IMPLEMENTASI *DYNAMIC DIFFICULTY ADJUSTMENT* PADA *RACING GAME* MENGGUNAKAN METODE *FUZZY*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Reza Saputra
NIM: 145150201111137

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI *DYNAMIC DIFFICULTY ADJUSTMENT* PADA *RACING GAME* MENGGUNAKAN METODE *FUZZY*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Reza Saputra
NIM: 145150201111137

Skripsi ini telah diuji dan dinyatakan lulus pada
31 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
M. Aminul Akbar, S.Kom., M.T.	Tri Afirianto, S.T., M.T.
NIK. 201607 891013 1 001	NIK. 201309 851213 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 6 Agustus 2018

Reza Saputra

NIM: 145150201111137

KATA PENGANTAR

Assalamualaikum warahmatullahi wabarakatuh.

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga laporan skripsi yang berjudul "Implementasi *Dynamic Difficulty Adjustment* Pada *Racing Game* Menggunakan Metode *Fuzzy* " dapat terselesaikan.

Selama pengerjaan laporan skripsi penulis mendapat banyak pelajaran dan pengalaman baru. Penulis juga banyak mengaplikasikan ilmu yang didapat selama menduduki bangku perkuliahan ke dala laporan skripsi ini. Penulis menyadari bahwa laporan skripsi ini tidak akan berhasil tanpa dukungan, doa, dan bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih dan rasa hormat yang sebesar-besarnya kepada :

1. Bapak M. Aminul Akbar, S.Kom., M.T., dan bapak Tri Afirianto, S.T, M.T., selaku dosen pembing skripsi penulis yang begitu sabar membimbing penulis juga memberikan arahan dengan sangat baik kepada penulis sehingga dapat menyelesaikan skripsi ini.
2. Orang tua dan keluarga besar saya yang tidak henti-hentinya memberikan doa dan dukungan selama proses menempuh pendidikan di Malang termasuk saat proses pengerjaan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si.,M.T., Ph.D., selaku Dekan Fakultas Ilmu Komputer.
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D., selaku Ketua Jurusan Teknik Informatika.
5. Bapak Agus Wahyu Widodo, S.T., M.Cs., selaku Ketua Program Studi Teknik Informatika.
6. Rezky Darmawan yang sudah membantu menjelaskan dasar-dasar logika *fuzzy* dan segala tahap yang diperlukan dalam prosesnya.
7. Teman-teman L-Tifam yang selalu memberi bantuan berupa saran, motivasi, doa, dan semangat mulai dari awal perkuliahan hingga menyelesaikan skripsi.
8. Seluruh civitas akademika Fakultas Ilmu Komputer yang selalu bantuan dalam proses penyelesaian skripsi ini.
9. Teman-teman keminatan MGM dan teman – teman Teknik Informatika Angkatan 2014 lainnya yang selalu berbagi ilmu dari awal perkuliahan sampai tahap penyelesaian skripsi.
10. Teman-teman Sektim yang selalu memberi bantuan saran, motivasi, doa, semangat, ilmu, serta pengalaman dalam mengerjakan skripsi.
11.

11. Teman-teman kos Andong 31 yang telah membantu memberi motivasi dan semangat dalam mengerjakan skripsi ini, dan segala pengalamannya semasa perkuliahan di Malang.
12. Semua pihak yang telah banyak membantu, berbagi ilmu dan pengalaman, serta memberikan dukungan selama proses pengerjaan skripsi yang tidak dapat penulis sebutkan satu per satu.

Penulis mengakui bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan, sehingga kritik yang membangun dan juga saran sangat penulis butuhkan. Akhir kata penulis berharap agar skripsi ini bisa membawa manfaat bagi semua pihak yang membaca dan menggunakannya.

Wassalamualaikum warahmatullahi wabarakatuh.

Malang, 6 Agustus 2018

Penulis

rezasaputra.kks@gmail.com

ABSTRAK

Reza Saputra, Implementasi Dynamic Difficulty Adjustment Pada Racing Game Menggunakan Metode Fuzzy

Dosen Pembimbing : M. Aminul Akbar, S.Kom., M.T., dan Tri Afirianto, S.T, M.T.

Pada beberapa *racing game* terdapat fitur untuk memainkan permainan secara kompetitif baik melawan pemain lain atau melawan Kecerdasan Buatan (KB) atau yang biasa disebut *bot.* Dalam penerapannya, sering terjadi perbedaan kemampuan yang jauh antara pemain dengan *bot.* Hal ini menyebabkan adanya rasa bosan jika kemampuan pemain jauh lebih baik dibandingkan dengan *bot,* dan akan menimbulkan rasa kegelisahan jika kemampuan pemain jauh lebih rendah jika dibandingkan dengan kemampuan *bot*. Dalam beberapa *racing game* terdapat pilihan untuk memilih tingkat kesulitan sebelum bermain, namun fitur ini dirasa masih kurang dapat mengimbangkan kemampuan pemain dengan *bot* karena seiring berjalannya waktu kemampuan pemain dapat meningkat, dan pemain baru juga cenderung tidak mengetahui tingkat kesulitan yang sesuai dengan kemampuannya. Untuk mengatasi masalah tersebut maka peneliti akan menerapkan *Dynamic Difficulty Adjustment* (DDA) menggunakan metode *Fuzzy* yang mampu menyesuaikan kemampuan *bot* dengan kemampuan pemain seiring berjalannya waktu. Pengujian DDA dilakukan dengan menguji kemampuan pemain dengan *bot* DDA. Hasil pengujian menunjukkan bahwa *bot* DDA mampu menyesuaikan perilakunya dengan kemampuan yang dimiliki oleh pemain Hasil pengujian menunjukkan bahwa *bot* DDA mampu menyesuaikan perilakunya dengan kemampuan yang dimiliki oleh pemain, yang mana pada pengujian untuk penguji handal diperoleh nilai rata-rata selisih jarak tempuhnya 2,35 meter, sedangkan untuk penguji pemula diperoleh nilai rata-ratanya adalah 4,49 meter.

Kata kunci: *Fuzzy*, *Dynamic Difficult*, *Dynamic Difficulty Adjustment, DDA*, *Game, Racing Game*

ABSTRACT

Reza Saputra, Implementation of Dynamic Difficulty Adjustment on Racing Game Using Metode Fuzzy

Supervisors : M. Aminul Akbar, S.Kom., M.T., dan Tri Afirianto, S.T, M.T.

Some racing games has a feature to play the game competitively against other player or against Artificial Intelligence (AI) or commonly called as bot. In its implementation, we tend to find that the ability between the player and the bot is far. This causes boredom if the player has much higher ability than the bot, and will produce anxiety if the player's ability is much lower when compared with the bot. In some racing games there is an option to choose the difficulty level before starting the game, but this feature is still considered less effective to balance the ability of player and bot, because the ability of players can increase as the time goes by, and new players tend to confused that they don't know what category of difficulty that suits to their abitility. To solve the problem, researcher will implement Dynamic Difficulty Adjustment (DDA) by using Fuzzy method that able to adjust the ability of the bot according to player's ability over time. DDA testing is done by playing and matching players with bots. Test results show that DDA bots are able to adjust their behavior with the player's ability, from the test we received that the average traveled distance for expert player is 2,35 meters, whereas the average traveled distance for beginner player is 4,49 meters.

Keywords : Fuzzy, Dynamic Difficult, Dynamic Difficulty Adjustment, DDA, Game, Racing Game

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB 1 PENDAHULUAN

1.1 Latar belakang

Seiring zaman berkembang perkembangan *video game* terus terjadi, salah satunya pada *racing game*. Menurut data dari bigfish*game*s.com (Big Fish Games, Inc., n.d.), *racing game* memiliki angka keminatan sebanyak 4% untuk wilayah Amerika, 10% untuk wilayah Eropa, dan 3% pada wilayah Jepang. Meskipun secara angka *racing game* masih diungguli oleh jenis *video game* lainnya, namun *racing game* masih banyak diminati oleh pemain dari kalangan pecinta olahraga dan adrenalin. Produksi terhadap racing *game* juga terus dilakukan hingga saat ini. *Racing game* sendiri merupakan suatu jenis dalam *video game* yang mensimulasikan pertandingan balapan sebagaimana halnya balapan dalam dunia nyata, pemain diharuskan untuk mengendarai kendaraan balap dan berkompetisi dengan pembalap lainnya untuk mencapai garis *finish* dengan menduduki posisi terdepan.

Perkembangan dalam *racing game* juga terus terjadi baik dalam hal grafis, suasana permainan yang lebih realistis, dan juga dalam hal kompetitif. Pada beberapa *racing game* terdapat fitur untuk memainkan permainan secara kompetitif baik melawan pemain lain ataupun melawan Kecerdasan Buatan (KB) atau yang biasanya disebut *bot*. Pengembangan permainan yang kompetitif tidak hanya terjadi pada *racing game* tetapi juga pada *video game* dengan jenis lainnya. Tujuan dibuatkannya permainan yang kompetitif ini adalah untuk mencapai unsur *fun* dan *challenging* yang harus dimiliki oleh suatu *video game* (Salen & Zimmerman, 2004). Dalam *video game*, banyak hal yang dapat mempengaruhi tingkat kepuasan yang diperoleh pemain. Salah satunya adalah tingkat kesulitan yang dialami oleh pemain baik dalam melawan pemain yang lain maupun melawan kecerdasan buatan. Csikszentmihalyi (Silva, et al., 2016), mengungkapkan bahwa ketika seseorang sedang melakukan suatu aktivitas dan menghayatinya, orang tersebut akan fokus kepada hal yang dikerjakan dan akan memengaruhi keadaan mentalnya. Hal inilah yang terjadi ketika seseorang sedang bermain *game*. Pemain yang memiliki tingkat kemampuan yang tinggi akan cenderung bosan atau tidak merasakan tantangan dalam bermain ketika dihadapkan pada lawan yang memiliki kemampuan yang rendah, sebaliknya ketika pemain yang kemampuannya rendah akan merasa stres dan frustasi ketika dihadapkan pada pemain yang memiliki kemampuan yang tinggi dikarenakan adanya perbedaan tingkat kemampuan yang jauh yang membuat pemain merasa jauh tertinggal atau dikalahkan dengan mudah. Untuk itu diperlukan sistem yang dapat mengimbangkan faktor kemampuan pemain dengan tingkat kesulitan yang ada dalam suatu *game*.

Untuk menjawab permasalahan di atas, terdapat beberapa solusi yang ada seperti menambahkan fitur yang secara eksplisit memperbolehkan pemain untuk memilih tingkat kesulitanan permainan yang sesuai dengan kemampuannya sebelum memulai permainan tersebut. Namun solusi ini masih kurang optimal

karena pemain diharuskan untuk memilih tingkat kesulitanannya sebelum bermain dikarenakan seiring berjalannya permainan tingkat kemampuan bermain dapat meningkat, dan pemain baru juga cenderung tidak mengetahui tingkat kesulitan yang sesuai dengan kemampuannya. Solusi lainnya adalah menggunakan ilmu Kecerdasan Buatan yaitu *Dynamic Difficulty Adjustment* (DDA). DDA dapat mengimbangkan kemampuan pemain dengan lawannya secara otomatis berdasarkan kemampuan pemain melalui komputasi cerdas yang dirancang oleh desainer dan pengembang. DDA banyak digunakan dalam mengatur tingkat kesulitan permainan dalam banyak *video game* seperti pada *Half-Life 2*, *Tower Defense Games,* dan lain-lain.

Sebelumnya terdapat penelitian yang mengangkat topik implementasi DDA menggunakan *Behavior trees* (BT) pada *Fighting Game* dengan jarak dan *delay* sebagai parameternya (Jacobsen, et al., 2011). BT merupakan suatu teknik Kecerdasan Buatan dalam *video game* yang mengatur tingkah laku yang dijalankan oleh suatu *agent* Kecerdasan Buatan berdasarkan beberapa *action* yang direpresentasikan menggunakan *Tree*. Kekurangan dari implementasi DDA menggunakan *Behavior trees* pada penelitian tersebut terdapat kesulitan dalam mengimplementasikannya karena menambahkan *utility* ke dalam framework BT terbukti merupakan hal yang sulit untuk dilakukan sebagaimana yang dialami oleh peneliti sebelumnya.

Penelitian lainnya yang diteliti oleh Rietveld (Rietveld, 2014) yaitu *circuit-adaptive rubber banding for racing games.* Penelitian ini memfokuskan untuk mengimbangkan kemampuan antar pemain dengan menyesuaikan jalur balap yang berbeda sesuai dengan kemampuan yang dimiliki oleh tiap pemain berdasarkan tingkat kemampuan *easy* dan *hard*. Kelebihan dari penelitian tersebut adalah hasilnya mampu meningkatkan keseimbangan pemain menjadi lebih kompetitif, namun memiliki kekurangan yang mana keluaran merupakan penyesuaian jalur yang berbeda untuk tiap pemain dirasa masih kurang adil.

Penulis mencoba untuk meningkatkan kekurangan yang terdapat pada DDA dalam *racing game* dengan mengimplementasikan metode *fuzzy*. Kelebihan dari metode *fuzzy*. Adapun kelebihan dari metode *fuzzy* adalah sangat fleksibel, konsep dasar yang mudah dimengerti, memiliki toleransi tinggi terhadap kebenaran suatu data, dan mampu membuat variasi dan mengambil langkah optimal dari suatu kondisi (Kusumadewi & Purnomo, 2010). Sebelumnya terdapat penelitian yang menggunakan metode *Fuzzy* untuk mengatur perilaku musuh dalam *Action-RPG Game* (Purba, et al., 2013). Pada hasilnya, penelitian mampu menerapkan *Fuzzy* ke dalam *Action-RPG Game* dan mampu menghasilkan keluaran sesuai dengan aturan yang ada dengan baik.

Dengan adanya penelitian DDA menggunakan metode *fuzzy* ini, diharapkan mampu meningkatkan keseimbangan antara pemain dengan *bot*, yang mana secara tidak langsung akan meningkatkan pengalaman pemain baik itu merasa lebih menyenangkan maupun menantang.

1.2 Rumusan masalah

Berdasarkan latarbelakang yang telah dipaparkan di atas, maka dapat dibuatkan rumusan masalah sebagai berikut:

1. Bagaimanakah cara untuk merancang dan mengimplementasikan DDA menggunakan metode *Fuzzy* pada *racing game*?
2. Bagaimanakah perbedaan hasil uji antara pemain pemula dengan pemain handal yang diperoleh dari penelitian?

1.3 Tujuan

Adapun tujuan dibuatkannya penelitian ini untuk mencapai beberapa hal seperti berikut:

1. Untuk merancang dan mengimplementasikan DDA menggunakan metode *Fuzzy* pada *racing game*.
2. Untuk mengetahui perbedaan dari hasil uji antara pemain pemula dengan pemain handal yang diperoleh dari penelitian.

1.4 Manfaat

Manfaat dari penelitian ini adalah untuk menghasilkan suatu metode yang dapat meningkatkan keseimbangan kemampuan pemain dengan Kecerdasan Buatan sesuai dengan kemampuan yang dimiliki oleh pemain. Adanya keseimbangan kemampuan antara pemain dengan KB juga akan meningkatkan pengalaman yang didapatkan oleh pemain, baik itu merasa lebih tertantang maupun menyenangkan.

1.5 Batasan masalah

Batasan masalah yang ditentukan dalam mengembangkan metode ini adalah sebagai berikut:

1. Penelitian ini dibatasi pada pengujian performa sistem dalam menyeimbangkan kemampuan pemain dengan kemampuan dari *bot*.
2. Penelitian ini diimplementasikan pada 3D *racing game.*

1.6 Sistematika pembahasan

Sistematika dalam penulisan yang digunakan dalam penelitian ini adalah sebagai berikut:

BAB I PENDAHULUAN
Menjelaskan mengenai latarbelakang, rumusan masalah, tujuan penelitian, manfaat dan batasan masalah.

BAB II LANDASAN KEPUSTAKA
Berisi paparan landasan-landasan teori dasar yang digunakan dalam penelitian, termasuk penelitian-penelitian sebelumnya.

BAB III METODOLOGI PENELITIAN
Menjelaskan tipe penelitian dan metode yang digunakan, objek penelitian dan teknik pengumpulan data dalam pengujian.

BAB IV PERANCANGAN
Menjelaskan perancangan parameter masukan dan keluaran yang digunakan dan juga merancang algoritme untuk setiap proses yang terdapat dalam metode *Fuzzy*.

BAB V IMPLEMENTASI
Menjelaskan bagaimana segala rancangan yang telah dibuat diterapkan ke dalam *game racing.*

BAB VI PENGUJIAN DAN ANALISIS
Melakukan pengujian terhadap *game racing* yang telah diterapkan metode DDA *Fuzzy* kemudian melakukan analisis terhadap hasil pengujian untuk mengetahui keberhasilan penelitian berdasarkan rumusan masalah.

BAB VI PENUTUP
Memaparkan kesimpulan dari hasil penelitian serta memberikan saran-saran yang mampu dilakukan selanjutnya untuk meningkatkan apa yang sudah dicapai pada penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Video game

Video game adalah suatu jenis permainan elektronik yang menggunakan suatu teknologi visual sebagai tampilan antarmuka seperti monitor dan dimainkan dengan tujuan untuk memperoleh hiburan. *Video game* mulai berkembang pada tahun 1970an (Gentile & Anderson, 2006). Permainan komersial pertama yang muncul adalah permainan Pong yang dikembangkan oleh Atari pada tahun 1972. Model *video game* terus berkembang dengan munculnya konsol-konsol seperti Magnavox Odyssey dan Atari Sears namun kurang menarik minat pasar. Perkembangan pesat terjadi pada tahun 1983 yang mana PC menjadi semakin canggih dan PC rakitan Commodore 64 mampu memainkan *video game*. Seiring berkembangnya teknologi, konsol-konsol baru muncul bersaing dipasar yang mana pada generasi ketiga kita mengenal Famicom Nintendo, pada generasi keempat terdapat NES dan Sega, generasi kelima didominasi oleh PlayStation dan juga beberapa saingannya seperti Sega Saturn dan Nintendo 64. Hingga pada saat ini teknologi *video game* banyak dimainkan pada PC dan juga beberapa konsol seperti PS4 dan XBOX-One. Beberapa jenis dasar *video game* adalah sebagai berikut (Arsenault, 2009):

a. *Action*, jenis permainan yang menekankan pada pergerakan fisik yang dikendalikan oleh koordinasi antara mata dan kemampuan motorik dari seorang pemain.

b. *Adventure*, jenis permainan yang menitikberatkan pemain untuk dapat menyelesaikan tantangan atau puzzle dengan cara berinteraksi dengan objek-objek yang ada didalam suatu permainan, dan jenis permainan ini biasanya dibawa dengan suatu narasi cerita sebagai petunjuk.

c. *Role-playing Game*, dalam jenis permainan ini, pemain mengendalikan suatu karakter yang memiliki *ability status/skill sets*, karakter akan berkembang seiring berjalannya permainan.

d. *Simulation*, jenis permainan yang menirukan perilaku atau pekerjaan yang terdapat didalam dunia nyata.

e. *Strategy*, jenis permainan yang menitikberatkan pada kemampuan otak untuk membuat rencana, strategi dan eksekusi yang diperlukan untuk memenangkan permainan.

f. *Sports*, jenis permainan yang mensimulasikan olahraga yang ada didalam dunia nyata.

g. *Driving*, jenis permainan yang mensimulasikan pemain untuk dapat mengendarai suatu alat transportasi dan saling bersaing untuk mencapai garis *finish*.

h. *Educational*, jenis permainan yang didalamnya terdapat unsur yang mendidik pemain serta memberikan wawasan baru.

2.2 Racing Game

Racing game pada dasarnya adalah gabungan dari jenis dari *Sports Game*. *Racing game* sendiri adalah jenis *game* yang mensimulasikan kompetisi balapan di dunia nyata, yang mana pemain berpartisipasi didalamnya bersaing melawan pembalap yang lain atau waktu (Drope, 2018). Pada dasarnya pemain menggerakkan suatu kendaraan balap yang akan bersaing dengan lawan balapnya yang lain dan diharuskan untuk mencapai garis akhir terlebih dahulu.

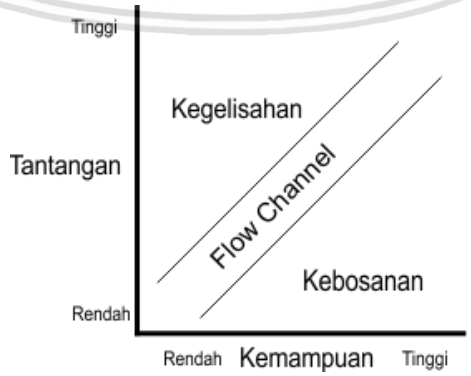
Racing game sendiri juga sudah dikembangkan pada awal 1970an. Permainan pertama yang tercatat sebagai *video game* dengan jenis balapan adalah *Space Race* yang dikembangkan oleh Atari (US Gamer, 2015).

Gambar 2.1 Video Game Atari Space Race Beserta Konsolnya
Sumber : (Skooldays.com, 2013)

Space Race dimainkan dalam konsol arcade dan menggunakan joystick untuk menggerakkan kendaraan. *Racing game* mengalami banyak perkembangan dan banyak diminati oleh pangsa pasar.

2.3 Psikologis dalam Bermain *Video Game*

Dalam bermain *video game*, terdapat parameter yang sangat penting yaitu kondisi psikologis yang dialami oleh pemain. Csikszentmihalyi membuatkan suatu model yang merepresentasikan keadaan psikologis tersebut seperti terlihat pada Gambar 2.2.



Gambar 2.2 *The Flow Theory*

Sumber : DDA Through Adaptive AI (2016)

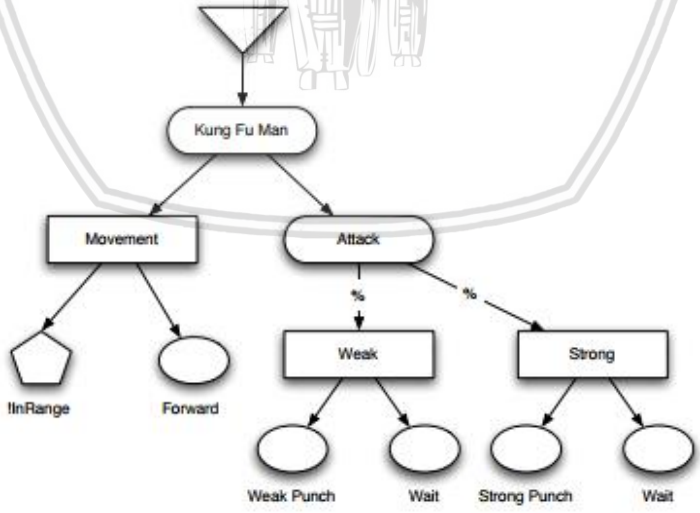
Berdasarkan Gambar 2.2, Csikszentmihalyi (Silva, et al., 2016) menjelaskan bahwa pemain yang memiliki tingkat kemampuan yang tinggi akan cenderung bosan atau tidak merasakan tantangan dalam bermain ketika dihadapkan pada lawan yang memiliki kemampuan yang rendah, sebaliknya ketika pemain yang kemampuannya rendah akan merasa stres dan frustasi ketika dihadapkan pada pemain yang memiliki kemampuan yang tinggi dikarenakan adanya perbedaan tingkat kemampuan yang jauh yang membuat pemain merasa jauh tertinggal atau dikalahkan dengan mudah.

*2.4 Dynamic Difficulty Adjustment (*DDA*)*

Dynamic Difficulty Adjustment merupakan suatu teknik yang digunakan dalam *video game* yang bertujuan untuk mengimbangkan tingkat kemampuan antar pemain dalam *game* tersebut dengan cara mengubah parameter, skenario dan perilaku yang ada dalam *game* (Silva, et al., 2016). Beberapa penelitian sebelumnya mengenai implementasi DDA ke dalam *game* adalah menggunakan *Behavior tree* dan *Circuit-adaptive Rubber Banding for Racing Games*.

2.4.1 DDA Menggunakan *Behavior tree*

Penelitian ini mengimplementasikan DDA dengan menggunakan metode *Behavior tree* ke dalam *game fighting* (Jacobsen, et al., 2011). *Behavior tree* merupakan suatu teknik Kecerdasan Buatan dalam *video game* yang mengatur tingkah laku yang dijalankan oleh suatu *agent* Kecerdasan Buatan berdasarkan beberapa *action* yang direpresentasikan menggunakan *Tree*. BT direpresentasikan menggunakan *node-node* yang dimulai dari *node root* kemudian dilanjutkan dengan beberapa *action* yang ada setelahnya. Contoh representasi *Behavior tree* untuk *Fighting Game* M.U.G.E.N. bisa dilihat pada Gambar 2.3.



Gambar 2.3 Contoh struktur *behavior tree*
Sumber : *Dynamic Difficulty Adjustment* Using *Behavior tree*s (2011)

Berawal dari *selector node* Kung Fu Man yang memiliki dua *child node* di antaranya *Movement* dan *Attack*. *Node* Kung Fu Man akan mengeksekusi *child node* pertama yaitu *Movement*, *node Movement* akan mengeksekusi *child node*

!InRange untuk melakukan pengkondisian apakah musuh berada dalam jarak yang ditentukan apa tidak, jika iya maka *node Forward* akan dieksekusi selanjutnya. Jika *node Movement* mengembalikan nilai salah, maka *child node Attack* akan dieksekusi. Pada *node Attack* akan diimplementasikan DDA menggunakan *Behavior tree* untuk mengetahui apakah musuh memiliki kemampuan *Weak* atau *Strong. Node Attack* memiliki *child node Weak* dan *Strong*, yang mana kedua *node* dieksekusi secara berbeda tergantung dari hasil yang diperoleh dari DDA. Jika keputusan DDA adalah *Weak,* maka *node Weak* akan dijalankan dan akan mengeksekusi *node Weak Punch* dan *Wait*, begitu pula jika keputusan DDA adalah *Strong,* maka *node Strong* akan dijalankan dan akan mengeksekusi *node Strong Punch* dan *Wait.*

2.4.2 Circuit-adaptive Rubber Banding for Racing Games

Pada penelitian ini, peneliti mencoba untuk menerapkan *Rubber Banding* ke dalam *racing game* dengan menyesuaikan jalur balapan sesuai dengan kemampuan pemain, *Rubber Banding* merupakan istilah lain yang sama dengan DDA (Rietveld, 2014). Peneliti menggunakan teknik *Support Vector Machine* (SVM) untuk mengklasifikasikan kemampuan pemain. SVM mengklasifikan data dengan cara mencari *hyperplane* yang mana *hyperplane* tersebut berfungsi untuk memisah dua buah kelas pada masukan *space.*

Setelah data kemampuan pemain diklasifikasikan, selanjutnya dipilihkan jalur dengan jenis-jenis yang berbeda yang sudah didefinisikan terlebih dahulu. Jalur-jalur tersebut dibedakan menyesuaikan dengan kemampuan pemain yang terdiri dari *easy* dan *hard.* Pemain dengan kemampuan yang tergolong *easy* akan diberikan jalur berjenis *easy* dan begitu juga sebaliknya untuk pemain dengan kemampuan *hard*.

Kesimpulan dari penelitian ini membuktikan bahwa hasil penelitian mampu mengimbangkan tingkat kompetisi antara pemain dengan kemampuan *easy* dengan *hard*, namun jenis keluaran yang dihasilkan dirasa masih kurang oleh penulis dikarenakan menggunakan jenis jalur yang berbeda untuk para pemain dan hal ini dirasa tidak adil.

Kelebihan dan kekurangan yang terdapat pada kedua metode penelitian di atas dipaparkan pada Tabel 2.1.

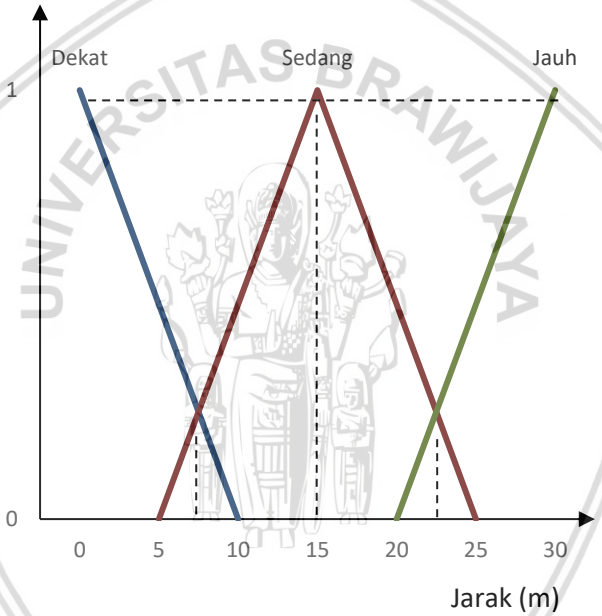
Tabel 2.1 Kelebihan dan Kekurangan Metode Penelitian Sebelumnya

No.	Penelitian	Kelebihan	Kekurangan
1	DDA menggunakan BT (Jacobsen, et al., 2011)	Dapat menghasilkan keputusan yang baik dan meningkatkan keseimbangan permainan dengan baik	Sulit diimplementasikan pada *video game* yang memiliki banyak *Utility*
2	*Circuit-adaptive Rubber Banding for Racing Games* (Rietveld, 2014)	Berhasil meningkatkan keseimbangan antar pemain secara signifikan	keluaran berupa pemilihan jalur yang sesuai dengan kemampuan pemain sehingga keseimbangan terlihat tidak adil

Dari kelebihan dan kekurangan Tabel 2.1, dapat dilihat bahwa DDA dapat ditingkatkan lagi dari segi kemudahan implementasi. Pada DDA menggunakan BT, peneliti mengatakan bahwa menambahkan *utility* (Cara mengukur dan membuat keputusan berdasarkan keluaran yang dihasilkan dari suatu kejadian) pada BT sulit untuk dilakukan. DDA juga dapat ditingkatkan dari segi bagaimana pemilihan keluaran tidak menganggu aspek *fun* yang harus dimiliki dalam suatu *game.*

2.5 Logika *Fuzzy*

Logika *Fuzzy* merupakan suatu cabang yang dipelajari dalam Kecerdasan Buatan (KB) yang dikembangkan oleh Dr. Lotfi Zadeh dari University of California pada tahun 1960an. Metode *Fuzzy* sendiri merupakan suatu perbaikan dari Logika Boolean. Pada Logika Boolean, ketika komputer dihadapkan pada kebenaran data maka akan menghasilkan nilai true(1) atau false(0), sedangkan pada Logika *Fuzzy* kebenaran suatu nilai ditetapkan dengan derajat kebenaran.



Gambar 2.4 Contoh Representasi Metode *Fuzzy* pada Pengukuran Jarak

Setiap variabel Dekat, Sedang, dan Jauh memiliki kebenaran nilai di ambang 0-1. Namun tiap-tiap variabel diberikan derajat kebenaran nilai. Contoh pada Dekat ketika jarak bernilai di bawah 5, maka jarak tersebut masih dikatakan Dekat, ketika jarak bernilai 8 maka nilai variabel Dekat berada di ambang 0 dan 1. Namun ketika tingkat suhu melewati *threshold* atau batas domain dari variabelnya yaitu 10, maka nilai dari variabel Dekat adalah 0.

Beberapa kelebihan yang terdapat dalam metode *Fuzzy* (Kusumadewi & Purnomo, 2010) :

a. Konsep logika yang mudah dimengerti.
b. Metode *fuzzy* fleksibel.
c. Memiliki toleransi terhadap kesalahan data.

d. Data-data dari para pakar dapat secara langsung dibangun dan diaplikasikan tanpa harus melalui proses *data training*.

e. Metode *fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional.

f. Penalaran didasarkan pada bahasa yang alami.

g. Mampu memetakan fungsi-fungsi nonlinear yang kompleks.

2.5.2 Himpunan Tegas (*Crisp Set*)

Himpunan yang keanggotaannya memiliki perbedaan kebenaran nilai yang jelas. Dalam himpunan tegas, nilai keanggotaan suatu elemen x dalam himpunan Y, memiliki dua kemungkinan, di antaranya :

a. Bernilai satu (1), menandakan bahwa elemen x merupakan anggota himpunan Y.
b. Bernilai nol (0), menandakan bahwa elemen x bukan merupakan anggota himpunan Y.

2.5.3 Himpunan *Fuzzy* (*Fuzzy Set*)

Himpunan yang keanggotaannya dinyatakan menggunakan derajat keanggotaan yang nilainya berada di antara 0 dan 1. Jika nilai suatu elemen x dalam suatu himpunan Y semakin mendekati satu (1), maka semakin tinggi kebenaran keanggotaan elemen x dalam himpunan Y. Jika nilai elemen x semakin mendekati 0, maka semakin rendah kebenaran keanggotaan elemen x dalam himpunan Y.

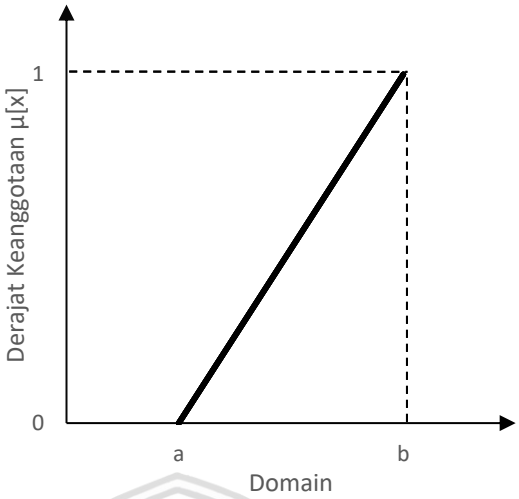
Himpunan Fuzzy terdiri dari dua atribut, yaitu :

a. Linguistik, merupakan nama yang digunakan dalam merepresentasikan kondisi atau keadaan untuk suatu kategori menggunakan bahasa alami. Contohnya seperti GELAP, CERAH, dan REDUP.
b. Numeris, yaitu angka yang merepresentasikan nilai dari suatu elemen atau variabel. Contohnya seperti 3, 100, 999, dan sebagainya.

2.5.4 Fungsi Keanggotaan (*Membership Function*)

Fungsi Keanggotaan adalah suatu kurva yang menunjukkan pemetaan derajat kebenaran keanggotaan suatu elemen dalam suatu himpunan (sering juga disebut dengan derajat keanggotaan) yang rentang nilainya berada antara 0 sampai 1. Untuk memperoleh nilai keanggotaan suatu elemen dalam suatu himpunan bisa dilakukan melalui pendekatan fungsi. Ada beberapa fungsi yang bisa digunakan di antaranya :

1. Fungsi Kurva Linear
   Berupa suatu garis lurus, terbagi menjadi menjadi dua, yaitu :
   a. Linear Naik, garis dimulai dari derajat 0 bergerak ke kanan menuju nilai domain dengan derajat yang lebih tinggi. Fungsi keanggotaan linear naik direpresentasikan seperti pada Gambar 2.5.



Gambar 2.5 Fungsi Keanggotaan Linear Naik

Persamaan keanggotaan linear naik:

$$\mu(x) = \begin{cases} 0; & x \leq a \\ \frac{x-a}{b-a}; & a \leq x \leq b \\ 1; & x \geq b \end{cases} \quad (2.1)$$

dimana,

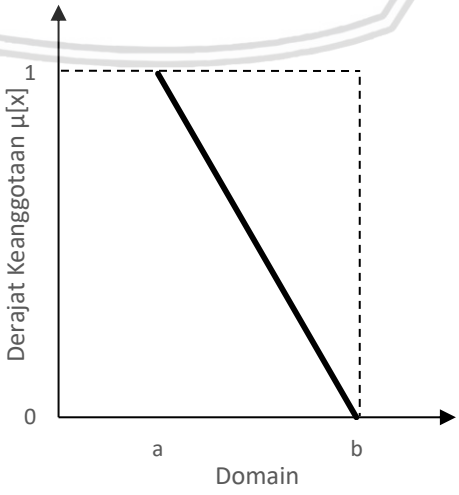
$\mu(x)$ = derajat keanggotaan variabel x

x = Nilai variabel

a = Domain derajat keanggotaan yang bernilai nol

b = Domain derajat keanggotaan yang bernilai satu

b. Linear Turun, dimulai dari derajat 1 pada sisi kiri bergerak ke kanan menuju nilai domain dengan derajat yang lebih rendah. Fungsi keanggotaan linear naik direpresentasikan seperti pada Gambar 2.6.



Gambar 2.6 Fungsi Keanggotaan Linear Turun

Persamaan keanggotaan linear turun :

$$\mu(x) = \begin{cases} 0; & x \geq b \\ \frac{b-x}{b-a}; & a \leq x \leq b \\ 1; & x \leq b \end{cases} \quad (2.2)$$

dimana,

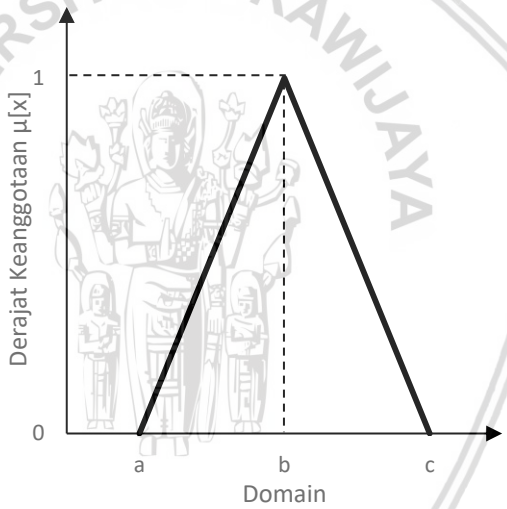
$\mu(x)$ = derajat keanggotaan variabel x

x = Nilai variabel

a = Domain derajat keanggotaan yang bernilai satu

b = Domain derajat keanggotaan yang bernilai nol

2. Fungsi Kurva Segitiga
   Membentuk segitiga, memiliki tiga titik parameter. Representasi fungsi keanggotaan keanggotaan segitiga dapat dilihat pada Gambar 2.7.



Gambar 2.7 Fungsi Keanggotaan Segitiga

Persamaan keanggotaan segitiga :

$$\mu(x) = \begin{cases} 0; & x \leq a \; atau \; x \geq c \\ \frac{x-a}{b-a}; & a \leq x \leq b \\ \frac{b-x}{c-b}; & b \leq x \leq c \end{cases} \quad (2.3)$$

dimana,

$\mu(x)$ = derajat keanggotaan variabel x

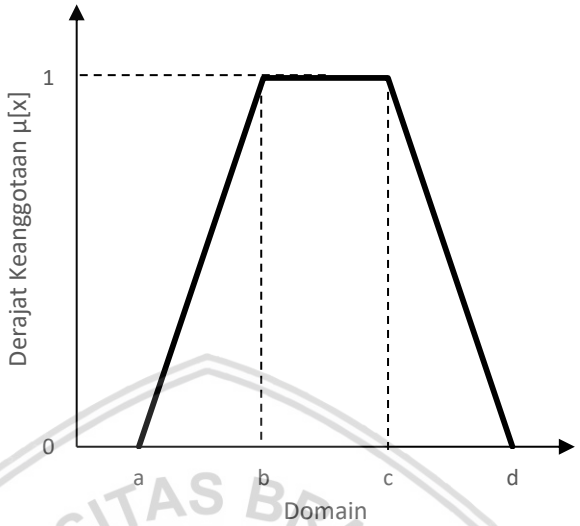
x = Nilai variabel

a = Domain terkecil derajat keanggotaan yang bernilai nol

b = Domain derajat keanggotaan yang bernilai satu

c = Domain terbesar derajat keanggotaan yang bernilai nol

3. Fungsi Kurva Trapesium
   Membentuk trapesium, terdiri dari empat parameter. Fungsi keanggotaan trapesium direpresentasikan seperti pada Gambar 2.8.



Gambar 2.8 Fungsi Keanggotaan Trapesium

Persamaan keanggotaan trapesium :

$$\mu(x) = \begin{cases} 0; & x \leq a \; atau \; x \geq d \\ \frac{x-a}{b-a}; & a \leq x \leq b \\ \frac{d-x}{d-c}; & c \leq x \leq d \\ 1; & b \leq c \end{cases} \quad (2.4)$$

dimana,

$\mu(x)$ = derajat keanggotaan variabel x

x = Nilai variabel

a = Domain terkecil derajat keanggotaan yang bernilai nol

b = Domain terkecil derajat keanggotaan yang bernilai satu

c = Domain terbesar derajat keanggotaan yang bernilai satu

d = Domain terbesar derajat keanggotaan yang bernilai nol

2.5.5 Operator Dasar untuk Operasi Logika *Fuzzy*

a. Operator AND (*Intersection*)
   Merupakan operator yang berhubungan dengan operasi irisan (∩) pada himpunan. Misalkan jika himpunan R adalah irisan dari himpunan P dan himpunan Q, maka dapat didefinisikan persamaan sebagai berikut:

$$\begin{aligned} R &= (P \cap Q)(x) \\ &= \min\{P(x), Q(x)\} \\ &= P(x) \cap Q(x), \forall x \in X \end{aligned} \quad (2.5.1)$$

Dengan derajat keanggotaannya adalah :

$$\mu_R(x) = \min(\mu_P(x), \mu_Q(x)) \\ = (\mu_P(x), \mu_Q(x)) \quad (2.5.2)$$

b. Operator OR (*Union*)
Merupakan operator yang berhubungan dengan operasi gabungan (∪) pada himpunan. Misalkan jika himpunan R adalah gabungan dari himpunan P dan himpunan Q, maka dapat didefinisikan persamaan sebagai berikut:

$$R = (P \cup Q)(x) \\ = \max\{P(x), Q(x)\} \\ = P(x) \cup Q(x), \forall x \in X \quad (2.6.1)$$

Dengan derajat keanggotaannya adalah :

$$\mu_R(x) = \max(\mu_P(x), \mu_Q(x)) \\ = (\mu_P(x), \mu_Q(x)) \quad (2.6.2)$$

2.5.6 Metode Sistem Inferensi *Fuzzy* Tsukamoto

Sistem Inferensi *Fuzzy* atau biasa disebut *Fuzzy Inference System* adalah serangkaian proses yang dilakukan untuk memperoleh nilai keluaran. Proses-proses yang dilalui untuk menalarkan suatu data dalam metode *Fuzzy* Tsukamoto adalah sebagai berikut (Fitri & Mahmudy, 2017) :

1. *Fuzzyfication*, dalam tahap ini nilai parameter masukan akan diubah dari nilai tegas menjadi variabel linguistik dengan menggunakan fungsi keanggotaan, kemudian dari proses tersebut akan dihasilkan nilai derajat keanggotaan.
2. Pembentukan *Rule* IF-Then, dalam tahap ini akan didefinisikan aturan-aturan yang dinyatakan dalam bentuk IF-Then yang nantinya digunakan dalam proses Inferensi.
3. Inferensi, dalam tahap ini akan dicari nilai α-predikat untuk setiap aturan yang ada ($\alpha_1, \alpha_2, \alpha_3, \ldots \alpha_n$) menggunakan fungsi implikasi MIN. Selanjutnya menghitung nilai z dengan persamaan berikut:

   $$z = a - ((a - b) * \alpha - predikat\ ) \quad (2.7)$$

   a = nilai domain maksimum, b = nilai domain minimum

   Nilai α-predikat adalah nilai minimal dari derajat keanggotaan, sedangkan nilai z merupakan nilai tegas untuk parameter keluaran.
4. *Defuzzyfication*, tahap akhir yang melakukan proses mengubah keluaran *fuzzy* yang diperoleh pada tahap inferensi kemudian diubah menjadi nilai *Center Average Defuzzyfier* atau nilai Z yang nantinya akan digunakan lebih lanjut dalam program. Nilai Z dihitung menggunakan persamaan berikut:

   $$Z = \frac{\sum \alpha - predikat_i * z_i}{\sum \alpha - predikat_i} \quad (2.8)$$

2.5.7 Implementasi Logika *Fuzzy* Untuk Mengatur Perilaku Musuh dalam Game Bertipe Action-RPG

Penelitian ini berfokus pada mengatur perilaku musuh dalam *action-rpg game* (Purba, et al., 2013). Perilaku musuh diatur menggunakan logika *Fuzzy* dengan menggunakan variabel *ammo, life, dan range* sebagai parameter masukan. Hasil perilaku yang diberikan adalah melarikan diri, bertahan, ragu-ragu, menyerang, dan menyerang brutal. Musuh yang terdapat memiliki jenis yang berbeda, terdiri dari penyerang, pemanah, dan boss. *Rule Fuzzy* didefinisikan untuk setiap jenis musuh.

Dalam penelitian tersebut dapat disimpulkan bahwa penerapan *Fuzzy* untuk mengatur perilaku musuh pada *action-RPG game* berhasil dan mampu memberikan hasil keputusan yang baik dan diharapkan dengan adanya hasil penelitian ini dapat digunakan untuk penulis dalam menunjang penelitian yang akan dikerjakan oleh penulis.

2.6 Alat Pengembangan

Paparan alat-alat pengembangan yang digunakan dalam penelitian, nantinya digunakan dalam tahap implementasi.

2.6.1 Unity *Game Engine*

Merupakan suatu aplikasi *game engine* buatan Unity Technologies yang membantu pengembangan *video game* dan dapat dijalankan pada banyak platform (Porsinelli, 2013). Unity dapat digunakan untuk mengembangkan *video game* dan simulasi pada platform PC, konsol, dan perangkat bergerak. Unity mendukung pengembangan *video game* berbasis grafis 3D dan 2D dan menggunakan teknologi grafis Direct3D dan OpenGL. Beberapa kelebihan Unity *Game Engine* adalah (Porsinelli, 2013):

a. Memiliki kemampuan pengembangan yang *mutiplatform.*
b. Memiliki *workflow* atau alur kerja yang baik.
c. Tampilan *Editor* yang ramah dan mudah untuk digunakan.
d. Memiliki fitur-fitur yang lengkap yang membantu pengembangan.
e. Perubahan dapat dilakukan secara langsung ketika *game* dalam keadaan berjalan atau *runtime*.

2.6.2 Visual Studio

Visual Studio adalah suatu alat bantu pemrograman terintegrasi atau biasa disebut *Integrated Development Environment* (IDE) yang dikembangkan oleh Microsoft Corporation. Visual Studio bisa didapatkan secara gratis menggunakan versi Community. Beberapa fitur utama yang terdapat pada Visual Studio di antaranya adalah (Wikipedia, 2018) :

a. *Code Editor*, yaitu fitur utama yang juga terdapat pada IDE pada umumnya. Berfungsi sebagai wadah untuk menuliskan serangkaian kode perintah yang diperlukan pada suatu program yang akan dikembangkan.

b. *Debugger*, fitur yang berfungsi untuk melakukan pengecekan terhadap kode-kode yang telah dituliskan. *Debugger* sangat berguna bagi pengembang karena dengan fitur ini mereka dapat melihat kesalahan yang ada dan segera melakukan perbaikan.

c. *Designer*, berfungsi untuk mendesain tampilan antarmuka program. Fitur ini mendukung proses desain dengan cara *drag and drop,* yang mana pengguna dapat menggeser objek atau *widget* yang tersedia ke dalam layar kerja.

d. Mendukung pemrograman dengan banyak bahasa. Beberapa bahasa yang sudah tersedia adalah C, C++, C++/CLI, VB.NET, C#, F#, dan TypeScript. Untuk pengembangan menggunakan bahasa lain bisa dilakukan dengan cara mengunduh modulnya secara terpisah.

2.6.3 *Racing Game Starter Kit* (RGSK)

Template Projek yang tersedia dalam Unity Asset Store. *Racing Game Starter Kit* merupakan template projek yang menyediakan asset-asset untuk membantu pengembangan *Racing Game*. Beberapa fitur utama yang ditawarkan di antaranya adalah (Intense Game, 2015) :

a. *Smart AI system*
b. *Replay system*
c. *Vehicle Customization system*
d. *Customizable AI behavior*
e. *Customizable vehicle physics*

Adapun beberapa parameter penting yang dapat digunakan sebagai parameter keluaran adalah sebagai berikut:

1. *Acceleration Sensitivity*, yaitu parameter yang digunakan sebagai acuan sensitifitas KB untuk meningkatkan akselerasi mobil.
2. *Acceleration Wander*, yaitu parameter yang digunakan sebagai acuan kapasitas akselerasi mobil.
3. *Brake Sensitivity*, yaitu parameter yang digunakan sebagai sensitifitas seberapa sering mobil menggunakan rem.
4. *Steer Sensitivity*, yaitu parameter yang digunakan sebagai acuan seberapa sensitif mobil dalam berbelok.
5. *Avoidance Sensitivity*, yaitu parameter yang digunakan sebagai acuan seberapa tangkas mobil dalam menghindari tabrakan dengan mobil lain.
6. *Max Wander Distance*, yaitu parameter yang digunakan sebagai jarak maksimal mobil diluar titik jalur ketika berjalan.
7. *Wander Rate*, parameter yang digunakan untuk mengatur seberapa sering mobil berjalan diluar titik jalur.

8. *Caution Speed*, parameter yang digunakan dalam mengkalkulasikan kecepatan utama ketika dalam keadaan *caution* berfungsi sebagai tingkat kewaspadaan KB terhadap belokan dan sebagainya.
9. *Overtake Strength*, parameter yang digunakan sebagai acuan seberapa kuat mobil tersebut mampu mengambil posisi mobil yang berada didepan.
10. *Nitro Probability*, parameter yang digunakan untuk mengatur seberapa besar kemungkinan *nitro* digunakan ketika berada dalam area pemicu.
11. *Caution Angle*, parameter untuk mengukur tingkat kelengkungan jalur untuk menentukan keadaan *caution*.

2.7 Difficulty Evaluation

Difficulty Evaluation (DE) adalah suatu metode pengujian yang terus dilakukan untuk mengetahui apakah terdapat perubahan perilaku yang dilakukan oleh objek yang diteliti (Silva, et al., 2016). Pada dasarnya DE terdiri dari tiga bagian yaitu :

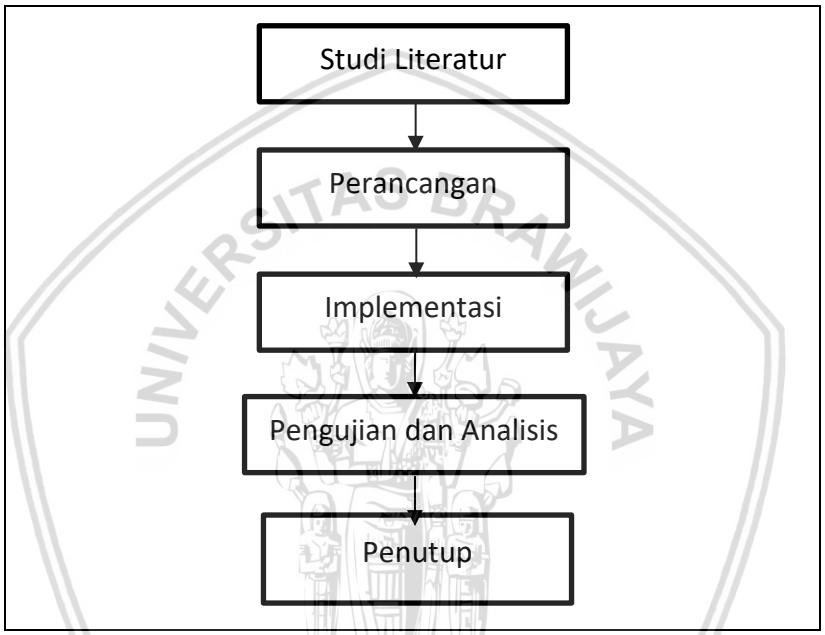
1. *Game Feature,* yaitu fitur atau variabel yang menjadi acuan dalam penelitian. Nilai dari suatu fitur tersebut akan diperoleh dan nantinya proses analisis akan dilakukan berdasarkan nilai-nilai dari fitur yang diperoleh.
2. *Difficulty Evaluation Proccess,* yaitu tahap yang mana proses analisis dilakukan untuk selanjutnya dilakukan penyesuaian parameter yang lebih baik. Adapun proses pengujiannya dibagi menjadi dua :
    a. *Baseline,* pengujian dilakukan dengan mengadu kemampuan semua *bot* statis dengan kemampuan yang berbeda, hal ini dilakukan untuk mengetahui kemampuan setiap *bot* dengan nilai parameter yang berbeda-beda.
    b. *Adaptive vs Static,* pengujian dilakukan dengan mengadu kemampuan *bot* dengan kemampuan yang statis dengan *bot* yang telah diimplementasi DDA. Dilakukan untuk mengetahui perbedaan peningkatan keseimbangan dari implementasi yang telah dilakukan.
3. *Dynamic Difficulty Adjustment Mechanism,* proses yang mana penyesuaian dilakukan, hal ini dilakukan untuk meningkatkan tingkat keseimbangan kemampuan antar objek pemain.

BAB 3 METODOLOGI

Dalam bab ini akan dijelaskan sistematika dari penelitian ini. Adapun proses menyelesaikan penelitian pada skripsi ini akan dilakukan dengan berdasarkan kerangka kerja serta dengan beberapa metode yang digunakan untuk menyelesaikan masalah.

3.1 Kerangka Kerja

Kerangka Kerja diperlukan untuk membantu mengarahkan penelitian ini. Adapun kerangka kerjanya direpresentasikan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Kerangka Kerja

3.2 Studi Literatur

Pada tahap ini dilakukan pengumpulan landasan teori yang diperoleh dari berbagai sumber seperti paper pendukung, internet, dan buku untuk melengkapi konsep dan teori-teori pendukung, sehingga nantinya penelitian memiliki landasan keilmuan yan baik. Adapun hasil studi literatur yang dikumpulkan untuk membantu landasan teori dalam penelitian ini adalah sebagai berikut:

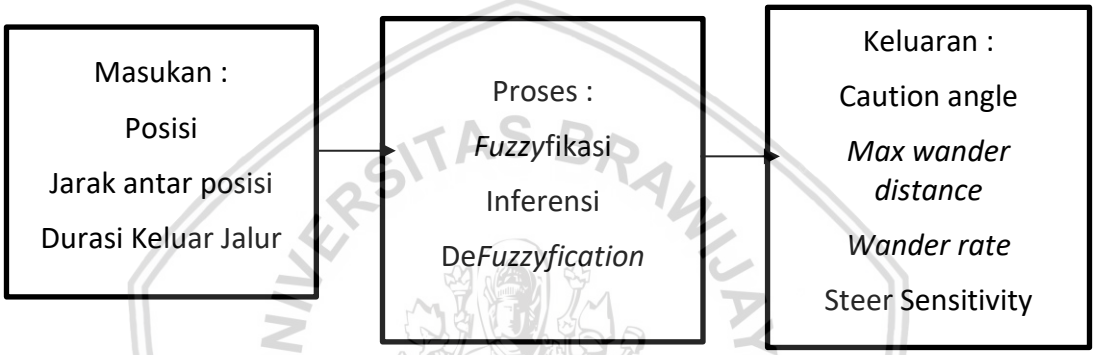
a. *Video Game*
b. *Racing game*
c. Psikologis dalam Bermain *Video Game*
d. *Dynamic Difficulty Adjustment*
e. Metode *Fuzzy*
f. Alat Pengembangan, meliputi :
    - Unity *Game* Engine
    - Visual Studio
    - *Racing Game Starter Kit*

3.3 Perancangan

Tahap perancangan merupakan tahap pengerjaan algoritme yang nantinya akan digunakan dalam penelitian. Secara garis besar, proses perancangan dibagikan menjadi dua bagian, yaitu :

1. Menentukan parameter-parameter masukan dan keluaran yang digunakan dalam proses *Fuzzy* nantinya.
2. Perancangan alur proses penerapan DDA menggunakan metode *fuzzy* ke dalam RGSK.
3. Perancangan algoritme DDA menggunakan metode *Fuzzy.*

Adapun rancangan proses *Fuzzy* yang dilakukan untuk memperoleh keluaran *crisp*-nya adalah sebagai berikut:



Gambar 3.2 Diagram Perancangan Proses *Fuzzy*

3.4 Implementasi

Tahap ini mengimplementasikan algoritme yang sudah dirancang sebelumnya dan kemudian diterapkan ke dalam *racing game*. Algoritme diimplementasikan ke dalam *Racing Game Starter Kit* menggunakan Unity *Game* Engine. Pemrograman dilakukan menggunakan Visual Studio IDE dengan bahasa pemrograman yang digunakan adalah CSharp.

Adapun proses implementasi yang akan dilakukan adalah sebagai berikut:

a. Mempersiapkan segala kebutuhan teknis dan fisik seperti alat-alat pengembang dan *hardware* yang digunakan dalam mengembangkan.

b. Mengimplementasikan algoritme DDA dengan metode *Fuzzy* ke dalam projek *game* ke dalam bentuk kode program berdasarkan rancangan yang telah dibuat.

3.5 Pengujian dan Analisis

Tahap ini merupakan tahap yang nantinya akan dilakukan untuk mengidentifikasi apakah dengan metode penelitian ini terdapat peningkatan dalam keseimbangan kemampuan antara pemain dengan *bot*. Metode pengujian yang nantinya akan digunakan oleh peneliti dalam mengambil hasil uji yang akan menjadi acuan dalam mengambil hasil akhir dari penelitian.

Objek yang diteliti adalah DDA pada *racing game*. Adapun pengujian dibagi menjadi dua jenis pengujian, yaitu :

1. Pengujian parameter *Fuzzy*, pengujian ini dilakukan untuk menguji seberapa besar pengaruh parameter yang digunakan akan berdampak pada hasil DDA.
2. Pengujian DDA, pengujian ini dilakukan untuk mengevaluasi apakah hasil keluaran yang diberikan oleh DDA yang telah diterapkan mampu memberikan hasil yang sesuai dengan tujuan penelitian, yaitu mengimbangkan kemampuan pemain dengan kemampuan *bot*.

Responden masing-masing terdiri dari satu orang pemain yang handal dan satu orang pemain pemula. Teknik pengumpulan data yang digunakan adalah dengan menguji para responden secara langsung untuk memainkan *racing game* yang menggunakan DDA dengan metode *Fuzzy* dan *racing game* tanpa menggunakan DDA dengan metode *Fuzzy*. Peneliti selanjutnya akan menganalisis hasil uji yang telah diperoleh dari para responden.

3.6 Penutup

Pada tahap ini akan dipaparkan kesimpulan yang diperoleh berdasarkan dari hasil penelitian, bagaimana hasil uji yang didapatkan dari rancangan dan implementasi yang telah dibuat. Kemudian pada tahap ini juga dipaparkan saran beberapa kekurangan yang mungkin masih dimiliki dalam penelitian ini dan dapat dikembangkan selanjutnya.

BAB 4 PERANCANGAN

4.1 Penentuan Parameter-parameter Masukan dan Keluaran Proses *Fuzzy*

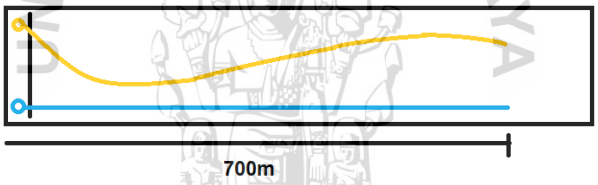
4.1.1 Penentuan Parameter Masukan *Fuzzy*

Dalam menentukan parameter-parameter masukan yang akan digunakan dalam proses *Fuzzy,* peneliti menentukan tiga parameter penting yang saling berpengaruh antara kemampuan yang dimiliki oleh pemain dengan hasil akhir balapan yang diperoleh. Adapun variabel-variabel yang akan digunakan sebagai parameter masukan dalam penelitian ini adalah sebagai berikut:

1. Posisi pemain
   Nilai parameter posisi diperoleh dengan cara mengambil nilai posisi sementara pemain pada saat waktu DDA *fuzzy* akan diimplementasikan.

   Persamaannya : $x = \frac{Posisi\ pemain}{Total\ Pebalap}$ (4.1.1)

2. Jarak pemain dengan lawan KB
   Untuk penjelasan bagaimana pengambilan jarak tempuh dapat dilihat pada ilustrasi Gambar 4.1.



Gambar 4.1 Ilustrasi Pengambilan Jarak Tempuh

Berdasarkan Gambar 4.1, garis berwarna biru melaju secara lurus tanpa mengalami banyak belokan, sedangkan garis berwarna kuning mengalami banyak belokan, nyatanya jarak yang ditempuh oleh garis kuning bisa melebihi 700 meter karena banyaknya belokan yang dialami, namun karena jarak yang diambil adalah jarak sebenarnya dari jalur lintasan, maka jarak yang diperoleh oleh garis kuning adalah 700 meter.

Nilai parameter jarak diperoleh dengan cara menghitung selisih jarak tempuh antara *bot* dengan pemain. Yang mana jarak tempuh merupakan seberapa jauh mobil telah melintasi lintasan balap yang dikalikan dengan *total laps*.

Persamaannya : $x = Abs(a - b)$ (4.1.2)

Yang mana a = jarak tempuh bot, b = jarak tempuh pemain, dan Abs adalah nilai absolut

3. Durasi mobil pemain berada diluar jalur (*offroad*)
   Nilai parameter *offroad* diperoleh dengan cara menghitung lamanya suatu bagian dari mobil *bot* ketika berada diluar jalur. Nilai parameter akan dikembalikan menjadi nol setiap kali DDA *fuzzy* akan diimplementasikan.

Peneliti menggunakan ketiga parameter tersebut dikarenakan sangat berpengaruh dalam menentukan kemampuan pemain dan nilai parameternya mudah untuk diperoleh.

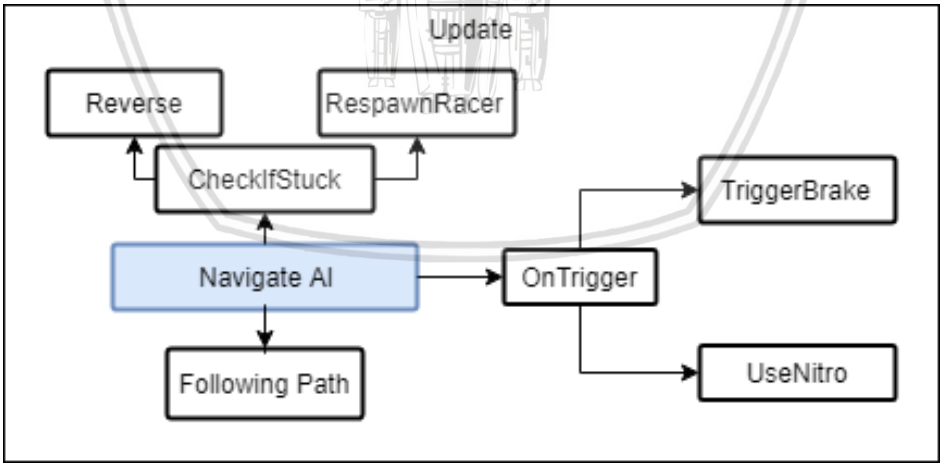
4.1.2 Penentuan Parameter Keluaran *Fuzzy*

Dalam menentukan parameter-parameter keluaran yang akan digunakan dalam proses *Fuzzy*, peneliti melihat ke dalam sisi kode program yang terdapat pada RGSK untuk mencari parameter-parameter penting yang mempengaruhi perilaku navigasi KB.

Peneliti menetapkan empat parameter yang akan digunakan dalam proses *Fuzzy* yaitu *caution angle*, *steer sensitivity*, *max wander distance*, dan *wander rate*. Peneliti memilih empat parameter tersebut sebagai parameter keluaran dikarenakan peneliti hanya ingin merubah kemampuan KB dengan memfokuskan pada kemampuan perilaku dari KB tersebut dan tanpa merubah kemampuan potensi dari mobil itu sendiri.

4.2 Rancangan DDA dengan Metode *Fuzzy*

Dalam penelitian ini DDA akan diterapkan ke dalam KB yang terdapat dalam *Racing Game Starter Kit*. *Racing Game Starter Kit* sudah menyediakan sistem navigasi KB namun masih menggunakan cara eksplisit untuk tingkat kesulitannya. Pemain dapat memilih tingkat kesulitan *easy, medium*, dan *hard* sebelum permainannya dimulai. Dalam sistem navigasi KB yang terdapat pada RGSK inilah nantinya akan diimplementasikan DDA dengan metode *Fuzzy*. Secara umum, alir kerja KB yang terdapat dalam RGSK tersebut ditunjukkan pada Gambar 4.2.

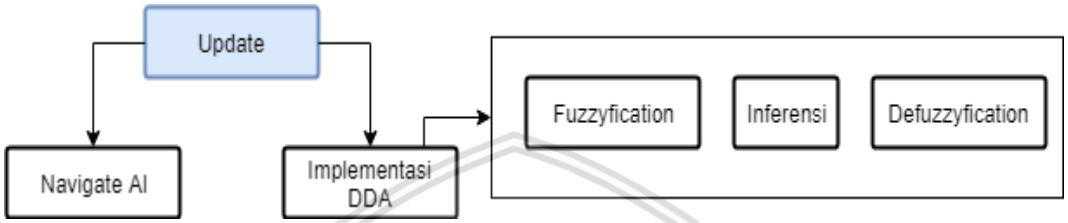


Gambar 4.2 Alir Kerja KB pada *Racing Game Starter Kit*

Navigate AI merupakan *node*/fungsi yang menjalankan perilaku AI secara umum. *Navigate AI* berada didalam *node* Update yang mana akan dijalankan secara terus-menerus selama balapan berjalan. Dari *node Navigate AI* menuju *Following Path* untuk menjalankan mobil *bot* mengikuti titik jalan yang terdapat pada lintasan. Kemudian terdapat *node CheckIfStuck* untuk melakukan pengecekan apakah mobil dalam keadaan terjebak atau berjalan melawan arah,

kemudian akan melakukan respawn atau reverse untuk mengembalikan mobil kembali pada keadaan normal. *Navigate AI* akan menjalankan *node OnTrigger* jika sewaktu-waktu mobil memasuki area *trigger,* yang mana *node UseNitro* atau *TriggerBrake* akan dijalankan tergantung dari jenis area yang dimasuki oleh mobil, adapun *UseNitro* adalah *node* untuk memerintahkan *bot* untuk menggunakan nitro (nos) dan *TriggerBrake* adalah *node* untuk memerintahkan *bot* untuk menggunakan rem.

Adapun rancangan untuk DDA menggunakan *Fuzzy* yang akan digunakan oleh peneliti digambarkan pada Gambar 4.3.



Gambar 4.3 Rancangan DDA dengan metode *Fuzzy*

Node Update merupakan fungsi di dalam *framework* (alat pengembang bantu) yang dimiliki oleh *Unity Game Engine, node Update* akan dijalankan secara terus menerus dalam delta waktu. Dalam rancangan yang telah digambarkan pada Gambar 4.3, *node Update* memiliki dua *child node* di antaranya *Navigate AI* dan Implementasi DDA. *Navigate AI* seperti yang dijelaskan sebelumnya merupakan fungsi yang menjalankan perilaku AI secara umum. *Node* Implementasi DDA merupakan fungsi yang mana nantinya DDA akan diimplementasikan setiap tujuh detik untuk memperoleh keluaran parameter *caution angle*, *steer sensitivity*, *max wander distance*, dan *wander rate* yang baru yang selanjutnya akan digunakan dalam *Navigate AI* dalam perulangan selanjutnya.

Dalam *node* Implementasi DDA akan terlebih dahulu menuju pada *node Fuzzyfication*, pada awalnya DDA akan mengambil nilai masukan berupa posisi, jarak, dan durasi *offroad* kemudian memasukkan nilai tersebut ke dalam proses *Fuzzyfication* untuk dicari derajat kebenarannya menggunakan fungsi keanggotaan untuk tiap masukan. Selanjutnya pada dari *node Fuzzyfication* dilanjutkan pada *node Inferensi. Node Inferensi* merupakan fungsi yang memproses pencarian nilai α-predikat dan z dari tiap rule yang telah didefinisikan untuk selanjutnya di proses pada *node Defuzzyfication*. Pada tahap *Defuzzyfication* akan dilakukan perhitungan untuk mencari nilai *weighted average* yang kemudian akan digunakan untuk menjadi nilai keluaran.

4.2.1 Parameter Masukan *Fuzzy*

Terdapat tiga parameter masukan yang digunakan dalam penelitian yaitu posisi, jarak, dan durasi offroad. Parameter masukan posisi yang diambil merupakan posisi dari pemain, parameter jarak yang diambil adalah jarak antar setiap KB dengan pemain, dan parameter durasi offroad merupakan durasi lamanya mobil pemain berada diluar jalur. Berikut merupakan fungsi *Fuzzy* dan grafik *Fuzzy* untuk setiap variabel masukan dalam penelitian ini:

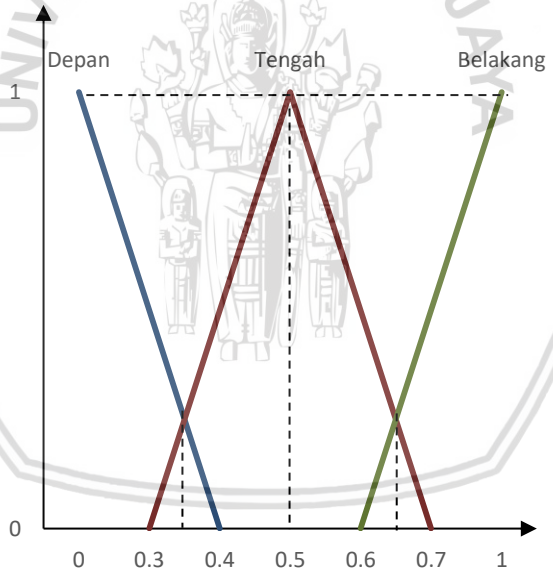
1. Posisi

Untuk parameter posisi dibagikan menjadi tiga kategori yaitu D, pertengahan dan terbelakang. Pemberian posisi direpresentasikan dengan nilai dengan skala 0-1 agar metode dapat digunakan dengan berapapun jumlah lawan KB yang dipilih. Himpunan *Fuzzy*nya dipaparkan pada Tabel 4.1 Himpunan Keanggotaan *Fuzzy* Variabel Posisi.

Tabel 4.1 Himpunan Keanggotaan *Fuzzy* Variabel Posisi

No	Kategori	Jarak
1	DEPAN	0 ≤ X ≤ 0,4
2	TENGAH	0,3 ≤ X ≤ 0,7
3	BELAKANG	0,6 ≤ X ≤ 1

Berdasarkan Tabel 4.1, suatu pembalap dikatakan berada pada kategori depan jika berada pada posisi dengan skala di bawah 0,4, sedangkan pembalap yang berada pada posisi dengan skala di antara 0,3 sampai 0,7 bisa dikatakan berada pada kategori tengah, dan pembalap dengan skala posisi di atas 0,6 bisa dikatakan berada pada kategori belakang. Adapun fungsi keanggotaan variabel posisi direpresentasikan pada Gambar 4.4.



Gambar 4.4 Representasi Himpunan *Fuzzy* Variabel Posisi

Berdasarkan Gambar 4.4, maka fungsi persamaan keanggotaan untuk himpunan DEPAN, TENGAH, dan BELAKANG dari variabel posisi dimodelkan sebagai berikut:

a. Fungsi persamaan himpunan posisi DEPAN diperoleh berdasarkan persamaan 2.2.

$$\mu PosisiDEPAN(x) = \begin{cases} 0; & x \geq 0{,}4 \\ \dfrac{0{,}4 - x}{0{,}4 - 0}; & 0 \leq x \leq 0{,}4 \end{cases} \quad (4.2.1)$$

b. Fungsi persamaan himpunan posisi TENGAH diperoleh berdasarkan persamaan 2.3.

$$\mu PosisiTENGAH(x) = \begin{cases} 0; & x \leq 0{,}3 \; atau \; x \geq 0{,}7 \\ \dfrac{x - 0{,}3}{0{,}5 - 0{,}3}; & 0{,}3 \leq x \leq 0{,}5 \\ \dfrac{0{,}7 - x}{0{,}7 - 0{,}5}; & 0{,}5 \leq x \leq 0{,}7 \end{cases} \quad (4.2.2)$$

c. Fungsi persamaan himpunan posisi Belakang diperoleh berdasarkan persamaan 2.1.

$$\mu PosisiBelakang(x) = \begin{cases} 0; & x \leq 0{,}6 \\ \dfrac{x - 0{,}6}{1 - 0{,}6}; & 0{,}6 \leq x \leq 1 \end{cases} \quad (4.2.3)$$

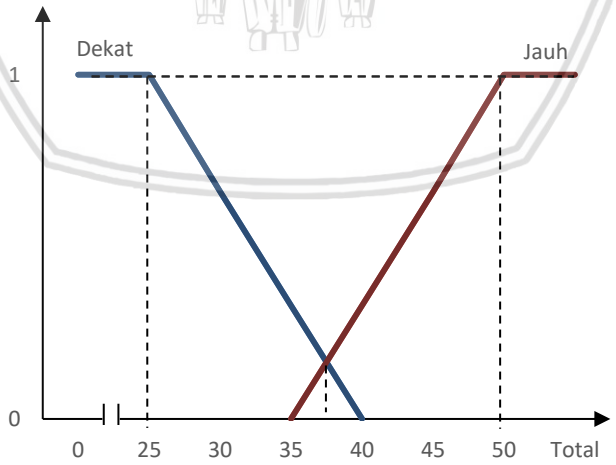
2. Jarak

Untuk parameter jarak dibagikan menjadi dua kategori yaitu dekat dan jauh. Parameter jarak diambil dari nilai rentang jarak antara KB dengan pemain. Himpunan *Fuzzy* jarak dipaparkan pada Tabel 4.2.

Tabel 4.2 Himpunan Keanggotaan *Fuzzy* Variabel Jarak

No	Kategori	Jarak
1	DEKAT	0 ≤ X ≤ 40 m
2	JAUH	35 ≤ X ≤ Panjang lintasan * Total Putaran

Berdasarkan Tabel 4.2, Perbedaan jarak bisa dimasukkan kategori dekat jika bernilai di antara 0 meter sampai dengan 40 meter, sedangkan perbedaan dimasukkan kategori jauh jika jaraknya bernilai lebih dari 35 meter. Adapun fungsi keanggotaan variabel jarak direpresentasikan pada Gambar 4.5.



Gambar 4.5 Representasi Himpunan *Fuzzy* Variabel Jarak

Berdasarkan Gambar 4.5, maka fungsi persamaan keanggotaan untuk himpunan DEPAN dan JAUH dari variabel jarak dimodelkan sebagai berikut:

a. Fungsi persamaan himpunan jarak DEKAT diperoleh berdasarkan persamaan 2.2.

$$\mu JarakDEKAT(x) = \begin{cases} 0; & x \geq 40 \\ \frac{40-x}{40-25}; & 25 \leq x \leq 40 \\ 1; & x \leq 25 \end{cases} \quad (4.3.1)$$

b. Fungsi persamaan himpunan jarak JAUH diperoleh berdasarkan persamaan 2.1.

$$\mu JarakJAUH(x) = \begin{cases} 0; & x \leq 28 \\ \frac{x-35}{50-35}; & 35 \leq x \leq 50 \\ 1; & x \geq 1 \end{cases} \quad (4.3.2)$$

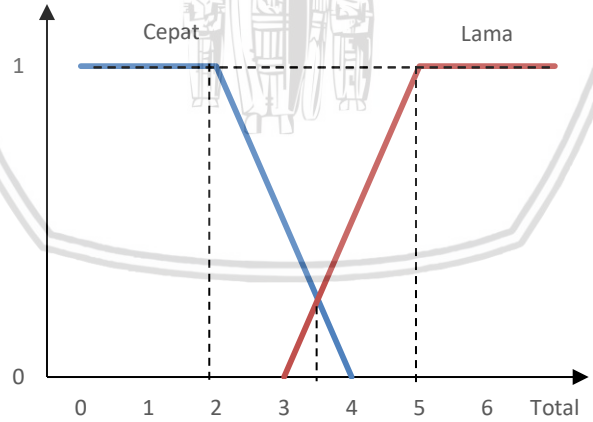
3. Durasi *Offroad*

Untuk parameter *offroad* dibagikan menjadi tiga kategori yaitu cepat dan lama. Parameter offroad merupakan lamanya durasi kendaraan pemain yang terhitung berada diluar jalur. Himpunan *Fuzzy* durasi *offroad* dipaparkan pada Tabel 4.3.

Tabel 4.3 Himpunan Keanggotaan *Fuzzy* Variabel Durasi Offroad

No	Kategori	Jarak
1	CEPAT	0 ≤ X ≤ 4
2	LAMA	3 ≤ X ≤ Total Waktu

Berdasarkan Tabel 4.3, durasi *offroad* dimasukkan ke dalam kategori cepat jika bernilai lebih kecil dari 4 detik dan kategori lama jika bernilai lebih dari 3 detik. Adapun fungsi keanggotaan variabel jarak direpresentasikan pada Gambar 4.6.



Gambar 4.6 Representasi Himpunan *Fuzzy* Variabel *Offroad*

Berdasarkan Gambar 4.6, maka fungsi persamaan keanggotaan untuk himpunan CEPAT dan LAMA dari variabel durasi *Offroad* dimodelkan sebagai berikut:

a. Fungsi persamaan himpunan durasi *Offroad* CEPAT diperoleh berdasarkan persamaan 2.2.

$$\mu OffCEPAT(x) = \begin{cases} 0; & x \geq 4 \\ \frac{4-x}{4-2}; & 2 \leq x \leq 4 \\ 1; & x \leq 2 \end{cases} \quad (4.4.1)$$

b. Fungsi persamaan himpunan durasi *Offroad* LAMA diperoleh berdasarkan persamaan 2.1.

$$\mu JarakJAUH(x) = \begin{cases} 0; & x \leq 2 \\ \dfrac{x-3}{5-3}; & 3 \leq x \leq 5 \\ 1; & x \geq 5 \end{cases} \quad (4.4.2)$$

4.2.2 Parameter Keluaran *Fuzzy*

Terdapat empat parameter keluaran yang nantinya akan diubah berdasarkan kondisi yang sesuai dari masukan. Terdiri dari parameter *caution angle, steer sensitivity, nitro probability, dan overtake strength*. Berikut merupakan fungsi *Fuzzy* dan grafik *Fuzzy* untuk setiap variabel keluaran dalam penelitian ini:

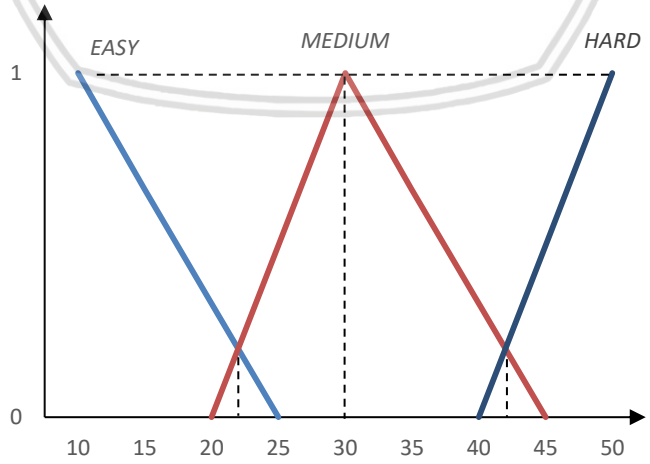
1. *Caution angle*

   Untuk parameter *caution angle* dibagikan menjadi tiga kategori yaitu *EASY*, *MEDIUM* dan *HARD*. Himpunan *Fuzzy* untuk variabel *caution angle* dipaparkan pada Tabel 4.4.

Tabel 4.4 Himpunan *Fuzzy* Variabel Caution angle

No	Kategori	Jarak
1	*EASY*	15 ≤ X ≤ 25
2	*MEDIUM*	20 ≤ X ≤ 45
3	*HARD*	40 ≤ X ≤ 50

Berdasarkan Tabel 4.4, *caution angle* dimasukkan ke dalam kategori *EASY* jika bernilai antara 15 sampai 25, kategori *MEDIUM* jika bernilai di antara 20 hingga 45, dan kategori *HARD* jika bernilai di antara 40 hingga 50. Adapun fungsi keanggotaan variabel caution angle direpresentasikan pada Gambar 4.7.



Gambar 4.7 Representasi Himpunan *Fuzzy* Variabel Caution angle

Berdasarkan Gambar 4.7, maka fungsi persamaan keanggotaan untuk himpunan *EASY*, *MEDIUM*, dan *HARD* dari variabel *caution angle* dimodelkan sebagai berikut:

a. Fungsi persamaan himpunan *caution angle* EASY diperoleh berdasarkan persamaan 2.2.

$$\mu caEASY(o1) = \begin{cases} 0; & x \geq 25 \\ \dfrac{25 - x}{25 - 10}; & 10 \leq x \leq 25 \end{cases} \quad (4.5.1)$$

b. Fungsi persamaan himpunan *caution angle* MEDIUM diperoleh berdasarkan persamaan 2.3.

$$\mu caMEDIUM(o1) = \begin{cases} 0; & x \leq 20 \; atau \; x \geq 45 \\ \dfrac{x - 20}{30 - 20}; & 20 \leq x \leq 30 \\ \dfrac{45 - x}{45 - 30}; & 30 \leq x \leq 45 \end{cases} \quad (4.5.2)$$

c. Fungsi persamaan himpunan *caution angle* HARD diperoleh berdasarkan persamaan 2.1.

$$\mu caHARD(o1) = \begin{cases} 0; & x \leq 40 \\ \dfrac{x - 40}{50 - 40}; & 40 \leq x \leq 50 \end{cases} \quad (4.5.3)$$

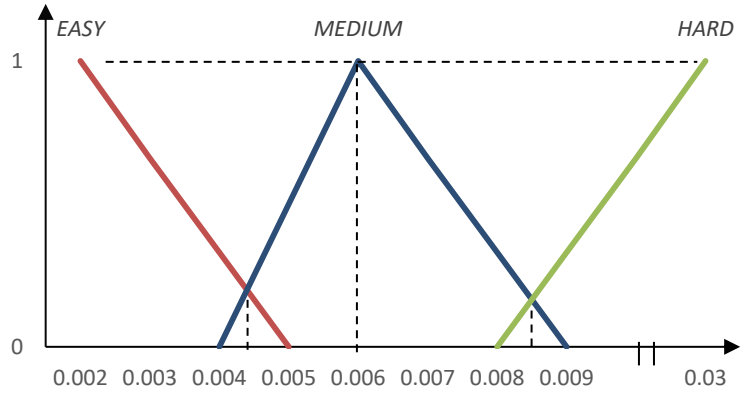
2. *Steer Sensitivity*

Untuk parameter *steer sensitivity* dibagikan menjadi tiga kategori yaitu *EASY*, *MEDIUM* dan *HARD*. Himpunan *Fuzzy steer sensitivity* dipaparkan pada Tabel 4.5.

Tabel 4.5 Himpunan *Fuzzy* Variabel *Steer Sensitivity*

No	Kategori	Jarak
1	*Easy*	0,002 ≤ X ≤ 0,005
2	*Medium*	0,006 ≤ X ≤ 0,009
3	*Hard*	0,008 ≤ X ≤ 0,03

Berdasarkan Tabel 4.5, steer sensitivity dimasukkan ke dalam kategori *EASY* jika bernilai di antara 0,002 sampai 0,005, kategori *MEDIUM* jika bernilai di antara 0,006 hingga 0,09, dan kategori *hard* jika bernilai di antara 0,008 hingga 0,03. Adapun fungsi keanggotaan variabel *steer sensitivity* direpresentasikan pada Gambar 4.8.



Gambar 4.8 Representasi Himpunan *Fuzzy* Variabel *Steer Sensitivity*

Berdasarkan Gambar 4.8, maka fungsi persamaan keanggotaan untuk himpunan *EASY*, *MEDIUM*, dan *HARD* dari variabel *steer sensitivity* dimodelkan sebagai berikut:

a. Fungsi persamaan himpunan *steer sensitivity EASY* diperoleh berdasarkan persamaan 2.2.

$$\mu ssEASY(o2) = \begin{cases} 0; & x \geq 0{,}05 \\ \dfrac{0{,}05 - x}{0{,}05 - 0.02}; & 0{,}02 \leq x \leq 0{,}05 \end{cases} \quad (4.6.1)$$

b. Fungsi persamaan himpunan *steer sensitivity MEDIUM* diperoleh berdasarkan persamaan 2.3.

$$\mu ssMEDIUM(o2) = \begin{cases} 0; & x \leq 0{,}004 \; atau \; x \geq 0{,}009 \\ \dfrac{x - 0{,}004}{0{,}006 - 0{,}004}; & 0{,}004 \leq x \leq 0{,}006 \\ \dfrac{0{,}009 - x}{0{,}009 - 0{,}006}; & 0{,}006 \leq x \leq 0{,}009 \end{cases} \quad (4.6.2)$$

c. Fungsi persamaan himpunan *steer sensitivity HARD* diperoleh berdasarkan persamaan 2.1.

$$\mu caHARD(o1) = \begin{cases} 0; & x \leq 0{,}008 \\ \dfrac{x - 0{,}008}{0{,}03 - 0{,}008}; & 0{,}008 \leq x \leq 0{,}03 \end{cases} \quad (4.6.3)$$

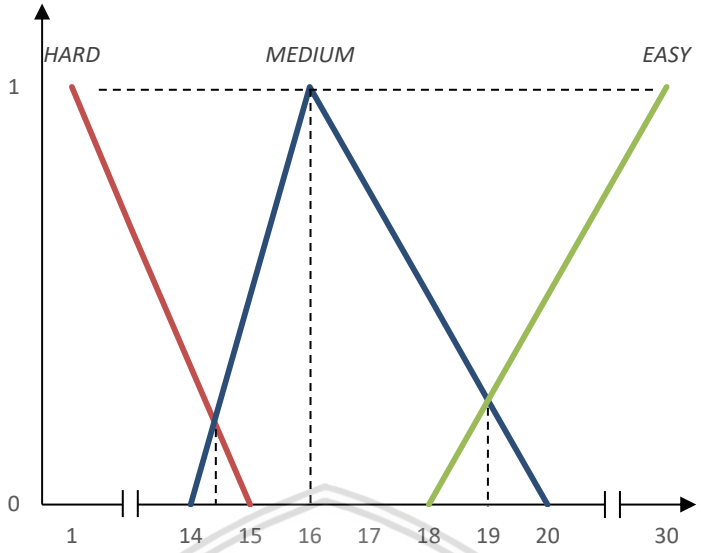
3. *Max Wander Distance*

Untuk parameter *max wander distance* dibagikan menjadi tiga kategori yaitu *EASY*, *MEDIUM* dan *HARD*. Himpunan *fuzzy max wander distance* dipaparkan pada Tabel 4.6.

Tabel 4.6 Himpunan *Fuzzy* Variabel *Max Wander Distance*

No	Kategori	Jarak
1	*EASY*	18 ≤ X ≤ 30
2	*MEDIUM*	14 ≤ X ≤ 20
3	*HARD*	1 ≤ X ≤ 15

Max wander distance dimasukkan ke dalam kategori *EASY* jika memiliki nilai jarak di antara 18 hingga 30, kategori *MEDIUM* jika memiliki nilai jarak di antara 14 hingga 20, dan kategori *HARD* jika memiliki nilai jarak di antara 1 hingga 15. Adapun fungsi keanggotaan untuk variabel *max wander distance* direpresentasikan pada Gambar 4.9.



Gambar 4.9 Representasi Himpunan *Fuzzy* Variabel *Max wander distance*

Berdasarkan Gambar 4.9, maka fungsi persamaan keanggotaan untuk himpunan *EASY*, *MEDIUM*, dan *HARD* dari variabel *max wander distance* dimodelkan sebagai berikut:

a. Fungsi persamaan himpunan *max wander distance EASY* diperoleh berdasarkan persamaan 2.1.

$$\mu mwEASY(o3) = \begin{cases} 0; & x \leq 18 \\ \dfrac{30-x}{30-18}; & 18 \leq x \leq 30 \end{cases} \quad (4.7.1)$$

b. Fungsi persamaan himpunan *max wander distance MEDIUM* diperoleh berdasarkan persamaan 2.3.

$$\mu mwMEDIUM(o3) = \begin{cases} 0; & x \leq 14 \; atau \; x \geq 20 \\ \dfrac{x-14}{16-14}; & 14 \leq x \leq 16 \\ \dfrac{20-x}{20-16}; & 16 \leq x \leq 20 \end{cases} \quad (4.7.2)$$

c. Fungsi persamaan himpunan *max wander distance HARD* diperoleh berdasarkan persamaan 2.2.

$$\mu mwHARD(o3) = \begin{cases} 0; & x \geq 15 \\ \dfrac{x-1}{15-1}; & 1 \leq x \leq 15 \end{cases} \quad (4.7.3)$$

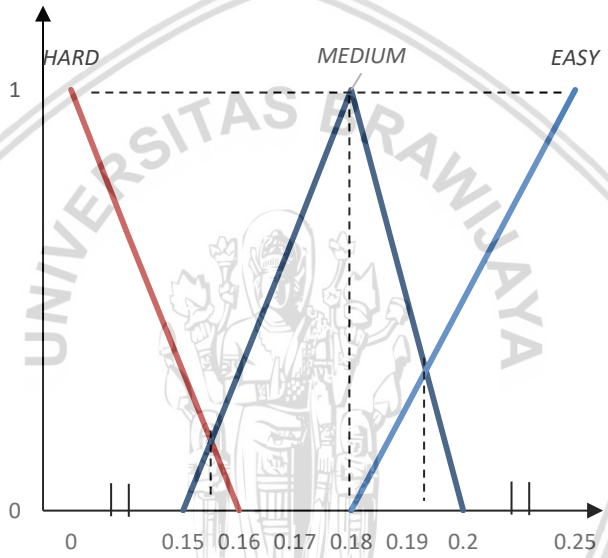
4. *Wander rate*

Untuk parameter *wander rate* dibagikan menjadi tiga kategori yaitu *EASY*, *MEDIUM* dan *HARD*. Himpunan *Fuzzy* untuk variabel *wander rate* dipaparkan pada Tabel 4.7.

Tabel 4.7 Himpunan *Fuzzy* Variabel *Wander rate*

No	Kategori	Range
1	*Easy*	0,18 ≤ X ≤ 0,25
2	*Medium*	0,15 ≤ X ≤ 0,2
3	*Hard*	0 ≤ X ≤ 0,16

Wander rate dimasukkan ke dalam kategori *EASY* jika bernilai antara 0,18 hingga 0,25, kategori *MEDIUM* jika bernilai di antara 0,15 hingga 0,2, dan kategori *HARD* jika bernilai di antara 0 hingga 0,16. Adapun fungsi keanggotaan variabel *wander rate* direpresentasikan pada Gambar 4.10.



Gambar 4.10 Representasi Himpunan *Fuzzy* Variabel *Wander rate*

Berdasarkan Gambar 4.10, maka fungsi persamaan keanggotaan untuk himpunan *EASY*, *MEDIUM*, dan *HARD* dari variabel *wander rate* dimodelkan sebagai berikut:

a. Fungsi persamaan himpunan *wander rate EASY* diperoleh berdasarkan persamaan 2.1.

$$\mu wrEASY(o4) = \begin{cases} 0; & x \leq 0{,}18 \\ \dfrac{0{,}25 - x}{0{,}25 - 0{,}18}; & 0{,}18 \leq x \leq 0{,}25 \end{cases} \qquad (4.8.1)$$

b. Fungsi persamaan himpunan *wander rate MEDIUM* diperoleh berdasarkan persamaan 2.3.

$$\mu wrMEDIUM(o4) = \begin{cases} 0; & x \leq 0{,}15 \; atau \; x \geq 0{,}20 \\ \dfrac{x - 0{,}15}{0{,}18 - 0{,}15}; & 0{,}15 \leq x \leq 0{,}18 \\ \dfrac{0{,}20 - x}{0{,}20 - 0{,}18}; & 0{,}18 \leq x \leq 0{,}20 \end{cases} \qquad (4.8.2)$$

c. Fungsi persamaan himpunan *wander rate HARD* diperoleh berdasarkan persamaan 2.2.

$$\mu mwHARD(o4) = \begin{cases} 0; & x \geq 0{,}16 \\ \dfrac{x-0}{0{,}16-0}; & 0 \leq x \leq 0{,}16 \end{cases} \quad (4.8.3)$$

4.2.3 Pendefinisian *Rule Fuzzy*

Rule Fuzzy merupakan bagian penting didalam logika *Fuzzy*. *Rule Fuzzy* berisi aturan-aturan yang nantinya digunakan untuk mengkategorikan masukan yang masuk ke dalam suatu himpunan *Fuzzy* yang telah didefinisikan dan kemudian menghasilkan keluaran yang sesuai berdasarkan kondisi dari masukan. Dalam penelitian ini, penulis merancang *Rule* seperti yang tertera pada Tabel 4.8.

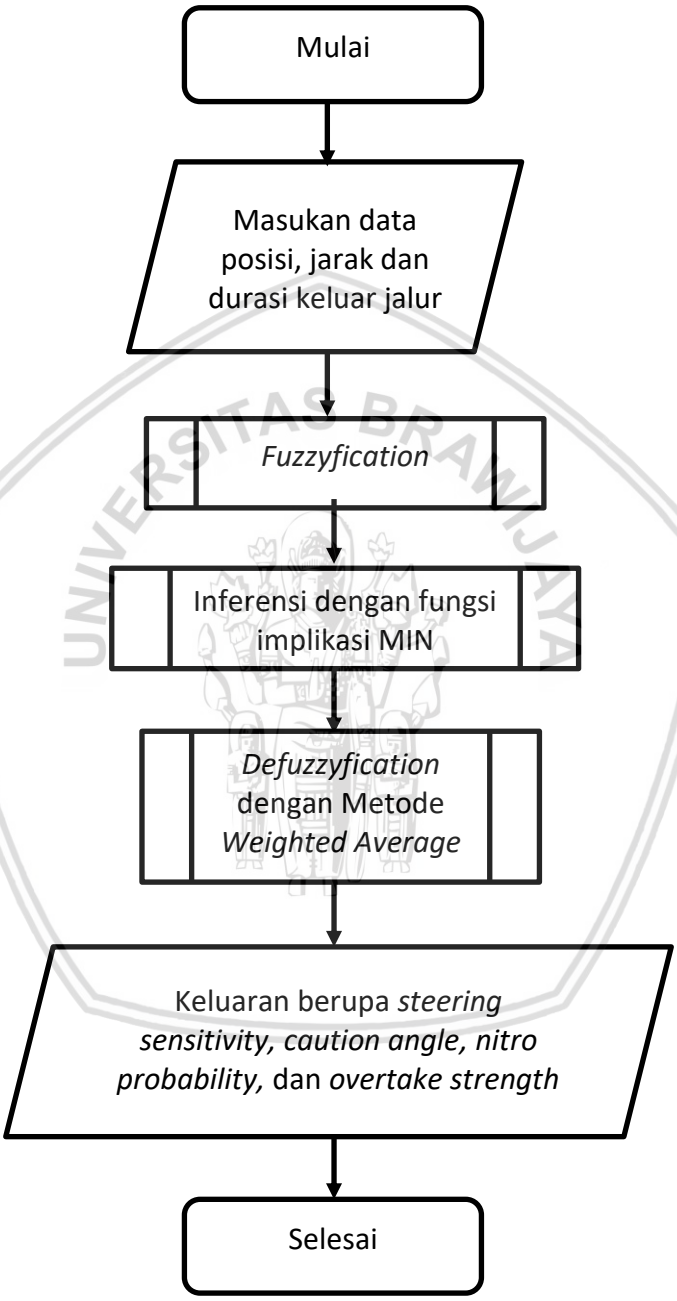
Tabel 4.8 Rancangan *Rule Fuzzy*

No	Masukan			Keluaran
	Posisi	Jarak	*Offroad*	
1	DEPAN	JAUH	CEPAT	*HARD*
2	DEPAN	JAUH	LAMA	*HARD*
3	DEPAN	DEKAT	CEPAT	*HARD*
4	DEPAN	DEKAT	LAMA	*MEDIUM*
5	TENGAH	JAUH	CEPAT	*MEDIUM*
6	TENGAH	JAUH	LAMA	*MEDIUM*
7	TENGAH	DEKAT	CEPAT	*MEDIUM*
8	TENGAH	DEKAT	LAMA	*MEDIUM*
9	BELAKANG	JAUH	CEPAT	*EASY*
10	BELAKANG	JAUH	LAMA	*EASY*
11	BELAKANG	DEKAT	CEPAT	*EASY*
12	BELAKANG	DEKAT	LAMA	*EASY*

Berdasarkan Tabel 4.8, jika posisi pemain masuk dalam kategori DEPAN, jarak mobil antara *bot* dan pemain masuk dalam kategori JAUH, dan durasi *offroad* masuk dalam kategori CEPAT, maka keluaran yang dihasilkan berupa *HARD,* yang mana keempat parameter keluaran (*caution angle, steer sensitivity, max wander distance, wander rate)* masing-masing akan masuk ke dalam kategori *HARD.* Begitupula jika parameter kondisi dari masukan menghasilkan keluaran berupa *MEDIUM*, keempat parameter keluaran akan masuk ke dalam kategori *MEDIUM*, begitu juga untuk keluaran kategori *EASY.*

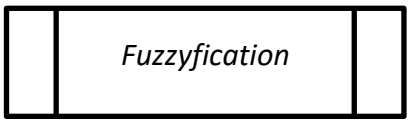
4.3 Rancangan Algoritme Metode *Fuzzy*

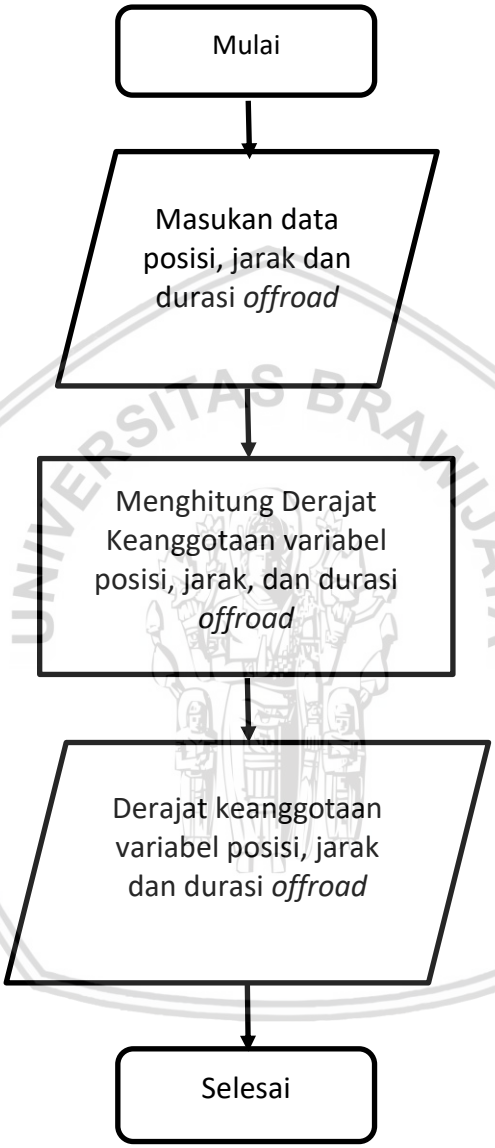
Pada tahap ini akan dijelaskan secara umum alir kerja yang dilalui dalam memperoleh hasil keluaran. Adapun untuk proses Inferensi, peneliti menggunakan metode Inferensi Tsukamoto. Diagram alir rancangan penerapan metode *Fuzzy* ditunjukkan pada Gambar 4.11.



Gambar 4.11 Diagram Alir Rancangan Metode *Fuzzy*

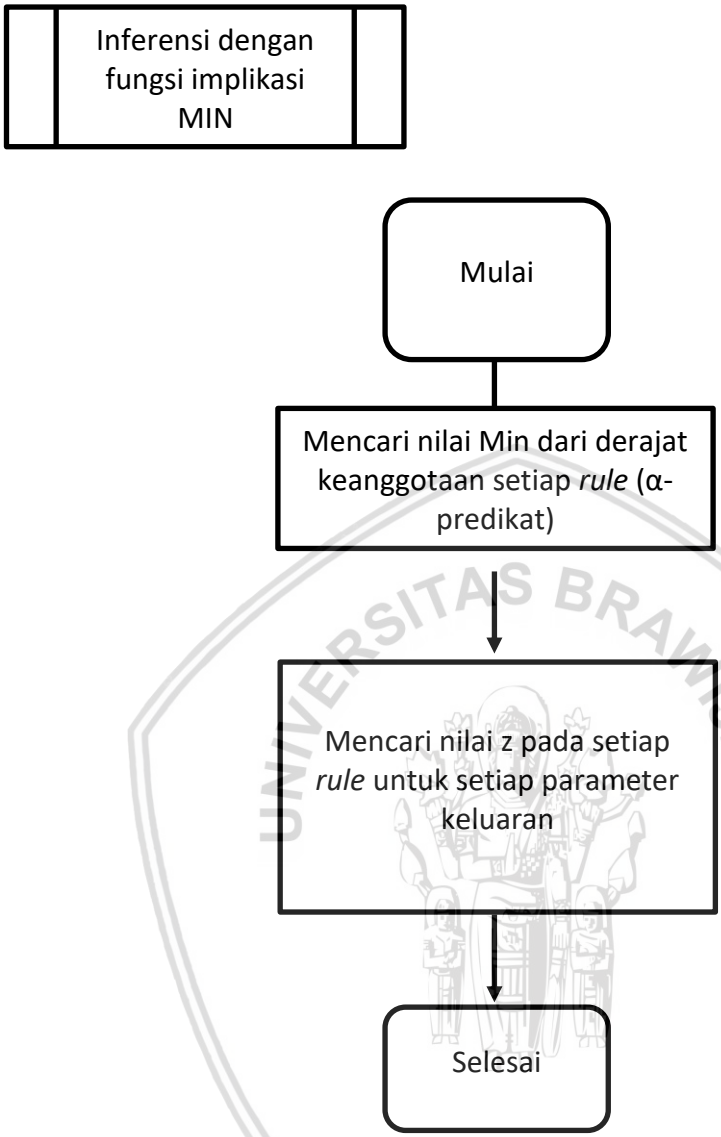
Berdasarkan Gambar 4.11, proses terbagi ke dalam tiga tahapan utama yaitu *Fuzzyfication*, inferensi dengan fungsi implikasi MIN, dan *Defuzzyfication* dengan metode *weighted average*. Pada proses *Fuzzyfication* akan dilakukan proses perhitungan derajat keanggotaan untuk masing-masing variabel masukan. Alur proses *Fuzzyfication* ditunjukkan pada Gambar 4.12.





Gambar 4.12 Diagram Alir Rancangan Tahap *Fuzzyfication*

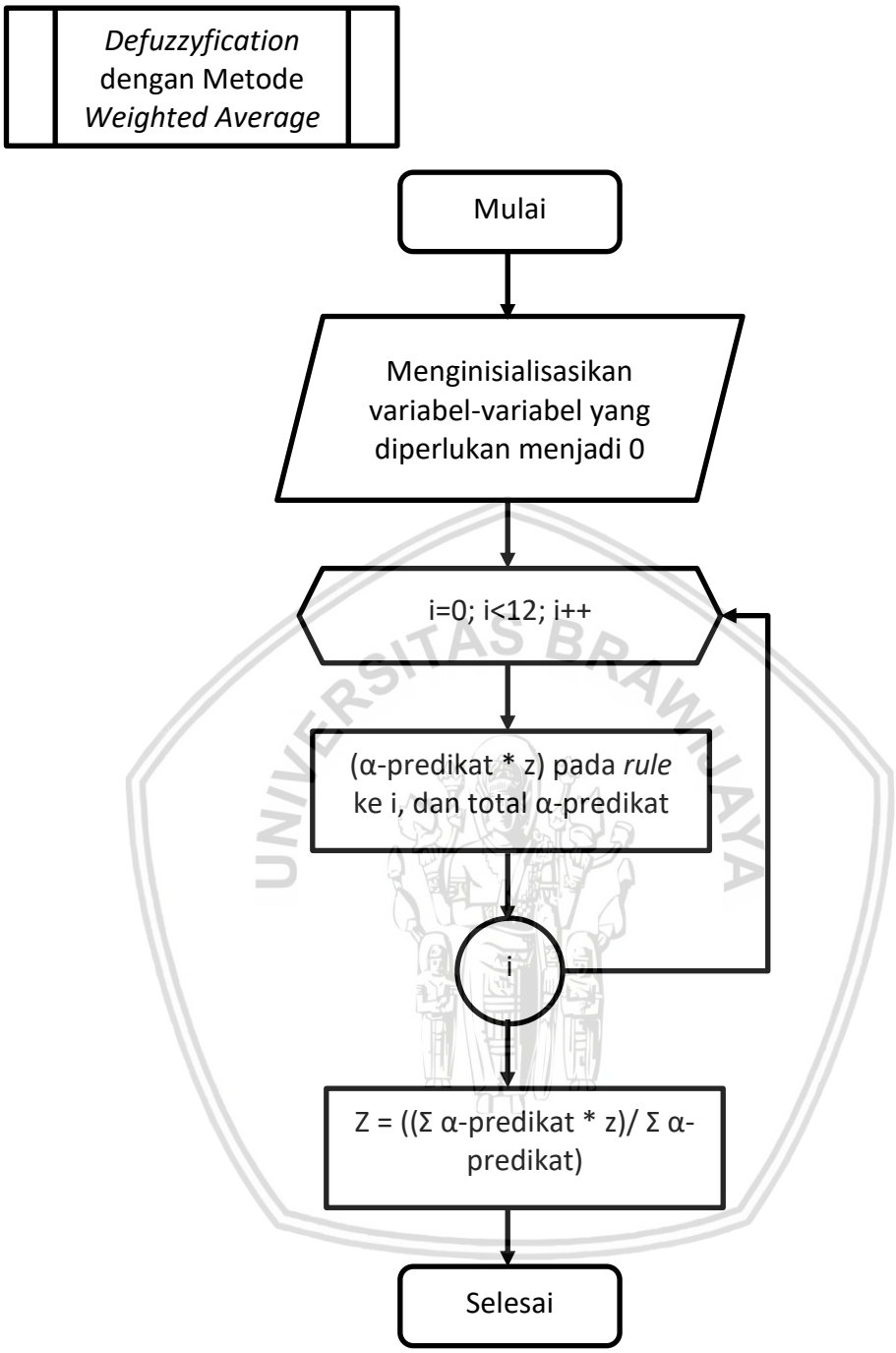
Berdasarkan Gambar 4.12, akan dicari derajat keanggotaan untuk setiap variabel masukan menggunakan persamaan *fuzzy* yang seperti yang telah dirancang pada subbab sebelumnya. Derajat keanggotaan dicari untuk setiap kategori karena diperlukan oleh *rule* yang telah didefinisikan untuk mencari nilai α-predikat pada tahap inferensi yang akan dilanjutkan selanjutnya. Untuk alur tahap Inferensi ditunjukkan pada Gambar 4.13.



Gambar 4.13 Diagram Alir Rancangan Tahap Inferensi

Setelah hasil derajat keanggotaan untuk setiap variabel diperoleh, kemudian dilanjutkan tahap inferensi yang bertujuan untuk mencari nilai α-predikat dan nilai z. Berdasarkan Gambar 4.13, untuk memperoleh nilai α-predikat maka dicari nilai minimum dari setiap derajat keanggotaan masing-masing rule. Nilai z diperoleh dari konsekuen dari setiap rule yang ada dengan menggunakan Persamaan 2.7.

Selanjutnya setelah nilai α-predikat dan nilai z sudah ditemukan maka akan dilanjutkan dengan proses *Defuzzyfication* untuk mencari nilai Z. Yang mana nilai Z adalah nilai akhir dari keluaran yang akan digunakan nantinya pada RGSK dalam perulangan selanjutnya. Untuk alur tahap *Defuzzyfication* ditunjukkan pada Gambar 4.14.



Gambar 4.14 Diagram Alir Tahap *Defuzzyfication*

Berdasarkan Gambar 4.14. Awalnya nilai variabel-variabel yang diperlukan seperti penyimpanan nilai sementara dan lain-lain menjadi 0. Kemudian melakukan perulangan sebanyak 12 kali perulangan(1 perulangan untuk satu rule). Didalam perulangan dilakukan perkalian antara α-predikat dengan z dan total α-predikat. Selanjutnya mencari nilai Z menggunakan Persamaan 2.8, yaitu membagi hasil perkalian α-predikat dengan z kemudian dibagi dengan total jumlah α-predikat. Setelah nilai Z diperoleh, kemudian nilai tersebut dimasukkan ke dalam variabel keluaran yang selanjutnya akan digunakan dalam RGSK pada perulangan berikutnya.

BAB 5 IMPLEMENTASI

5.1 Spesifikasi Lingkungan

Perangkat keras dan alat pengembangan yang digunakan dalam penelitian dirincikan sebagai berikut:

1. Laptop Asus a455ln

   Spesifikasi :

   - Sistem Operasi Windows 10 Pro
   - Tipe Sistem 64-bit
   - Prosesor Intel Core i5-4210U @1.70 GHz kapasitas 8GB
   - GPU Nvidia 840m kapasitas 2GB

2. *Unity Game Engine* versi 2017.1.0f3 (64-bit) *free edition*
3. *Racing Game Starter Kit* versi 1.1.0a
4. *Visual Studio Community 2015*

5.2 Implementasi Algoritme DDA dengan Metode *Fuzzy* pada RGSK

Pada tahap implementasi DDA dengan metode *Fuzzy* pada RGSK, algoritme yang telah dirancang akan diterapkan ke dalam projek RGSK. Seperti yang dipaparkan pada Gambar 4.3. DDA akan diimplementasikan dalam fungsi *Update* setiap tujuh detik. Kode sumber untuk implementasi DDA pada fungsi *Update* dipaparkan pada Kode Sumber 5.1.

Kode Sumber 5.1 Implementasi DDA Fuzzy

```
1   void Update()
2      {
3          CheckIfStuck();
4             if (RaceManager.instance.raceStarted == true)
5             {
6                if (period >= 7f)
7                {
8                   if (isFuzzy == true && tag == "Opponent")
9                   {
10                      //Debug.Log("implementing fuzzy");
11                      implement_DDA_Fuzzy();
12                      period = 0;
13                   }
14                }
15                period += UnityEngine.Time.deltaTime;
16             }
17      }
```

Berdasarkan Kode Sumber 5.1, pada baris 4 akan dilakukan pengkondisian apakah balapan sudah berjalan, jika iya maka akan dijalankan kode pada baris ke-6 untuk melakukan pengkondisian apakah waktu sudah berjalan tujuh detik, jika iya maka akan dijalankan kode pada baris 8 untuk melakukan pengkondisian

apakah isFuzzy bernilai true atau tag bernilai “Opponent”, jika benar maka akan fungsi implement_DDA_Fuzzy akan dijalankan untuk mengimplementasikan DDA *Fuzzy* dan kemudian nilai period akan diulang menjadi 0. Pada baris 15 merupakan kode untuk menambahkan nilai period dengan delta time. Selanjutnya kode untuk fungsi implement_DDA_Fuzzy dipaparkan pada Kode Sumber 5..

Kode Sumber 5.2 Proses implement_DDA_Fuzzy

```
1  Void implement_DDA_Fuzzy()
2      {
3          jarak = Mathf.Abs(player.GetComponent<ProgressTracker>()
4  .progressDistance - progTracker.progressDistance);
           posisiPemain = (float)
5  player.GetComponent<Statistics>().rank/totalRacer;
           offRoadDuration = Wheels.offRoadDuration;
6          fuzzyfication(posisiPemain, jarak, offRoadDuration);
7          inference();
8          defuzzyfication();
9      }
```

Berdasarkan Kode Sumber 5., akan dimasukkan nilai parameter masukan jarak, posisi dan durasi *offroad* seperti yang tertera pada kode baris 3,6 dan 8. Selanjutnya akan dijalankan fungsi *fuzzyfication, inference,* dan *defuzzyfication* secara berurut.

Kode Sumber 5.3 Proses *Fuzzyfication*

```
1  void fuzzyfication(float posisiPemain, float jarak, float
   offRoadDuration)
3      {
4          float pos_max = RankManager.instance.totalRacers;
5          WaypointCircuit circuit =
   FindObjectOfType<WaypointCircuit>();
6          position_membership_degree(posisiPemain);
7          distance_membership_degree(jarak);
8          offroad_membership_degree(offRoadDuration);
9      }
```

Berdasarkan Kode sumber 5.3, pada baris 6,7, dan 8 akan dijalankan fungsi position_membership_degree, distance_membership_degree, dan offroad_membership_degree untuk mencari derajat keanggotaan untuk parameter masukan. Kode untuk ketiga fungsi tersebut dipaparkan pada Kode sumber 5.4, Kode sumber 5.6, dan Kode sumber 5.7.

Kode Sumber 5.4 Mencari Derajat Keanggotaan *Offroad*

```
1  Void offroad_membership_degree(float offroad, float offroad_max)
2      {
3          float batas_kiri_cepat = 2;
4          float batas_kanan_cepat = 4;
5          float batas_kiri_lama = 3;
6          float batas_kanan_lama = 5;
7
8          //membership cepat
9          if (offroad <= batas_kiri_cepat)
10         {
11             offroad_cepat = 1;
```

```
12          }
13          if (offroad >= batas_kiri_cepat && offroad <=
14  batas_kanan_cepat)
15          {
16              offroad_cepat = ((batas_kanan_cepat - offroad) /
17  (batas_kanan_cepat - batas_kiri_cepat));
18          }
19          if (offroad >= batas_kanan_cepat)
20          {
21              offroad_cepat = 0;
22          }
23
24          //membership lama
25          if (offroad <= batas_kiri_lama)
26          {
27              offroad_lama = 0;
28          }
29          if (offroad >= batas_kiri_lama && offroad <=
30  batas_kanan_lama)
31          {
32              offroad_lama = ((offroad - batas_kiri_lama) /
33  (batas_kanan_lama - batas_kiri_lama));
34          }
35          if (offroad >= batas_kanan_lama)
36          {
37              offroad_lama = 1;
38          }
39      }
```

Berdasarkan Kode Sumber 5., baris 3-6 adalah perintah untuk mendeklarasikan dan menginisialisasikan variabel yang digunakan sebagai titik *fuzzy*. Selanjutnya pada baris 8-22 adalah perintah untuk mencari derajat keanggotaan kategori CEPAT berdasarkan Persamaan 4.4.1, sedangkan pada baris 24-38 adalah perintah untuk mencari derajat keanggotaan kategori LAMA berdasarkan Persamaan 4.4.2.

Kode Sumber 5.5 Mencari Derajat Keanggotaan Posisi

```
1           void position_membership_degree(float position, float
    pos_max)
2       {
3           float batas_kiri_depan = 0;
4           float batas_kanan_depan = 0.4f;
5
6           float batas_kiri_medium = 0.3f;
7           float tepi_medium_medium = 0.5f;
8           float batas_kanan_medium = 0.7f;
9
10          float batas_kiri_belakang = 0.6f;
11          float batas_kanan_belakang = 1;
12          //membership posisi depan
13          if (position <= batas_kiri_depan)
14          {
15             pos_depan = 1;
16          }
17          else if (position >= batas_kiri_depan && position <=
    batas_kanan_depan)
18          {
```

```
19              pos_depan = ((batas_kanan_depan - position) /
    (batas_kanan_depan - batas_kiri_depan));
20          }
21           else if (position >= batas_kanan_depan)
22          {
23               pos_depan = 0;
24          }
25
26          //membership posisi medium
27          if (position <= batas_kiri_medium || position >=
    batas_kanan_medium)
28          {
29              pos_medium = 0;
30          }
31          if (position >= batas_kiri_medium && position <=
    tepi_medium_medium)
32          {
33              pos_medium = ((position - batas_kiri_medium) /
    (tepi_medium_medium - batas_kiri_medium));
34          }
35          if (position >= tepi_medium_medium && position <=
    batas_kanan_medium)
36          {
37              pos_medium = ((batas_kanan_medium - position) /
    (batas_kanan_medium - tepi_medium_medium));
38          }
39
40          //membership posisi belakang
41          if (position <= batas_kiri_belakang)
42          {
43              pos_belakang = 0;
44          }
45          else if (position >= batas_kiri_belakang && position <=
    batas_kanan_belakang)
46          {
47              pos_belakang = ((position - batas_kiri_belakang) /
    (batas_kanan_belakang - batas_kiri_belakang));
48          }
49          else if (position >= batas_kanan_belakang)
50          {
51              pos_belakang = 1;
52          }
53      }
```

Berdasarkan Kode Sumber 5., baris 3-11 adalah perintah untuk mendeklarasikan dan menginisialisasikan variabel yang digunakan sebagai titik *fuzzy*. Selanjutnya pada baris 12-24 adalah perintah untuk mencari derajat keanggotaan kategori DEPAN berdasarkan Persamaan 4.2.1. Baris 26-38 adalah perintah untuk mencari derajat keanggotaan kategori LAMA berdasarkan Persamaan 4.2.2. Baris 40-52 adalah perintah untuk mencari derajat keanggotaan kategori BELAKANG berdasarkan Persamaan 4.2.3.

Kode Sumber 5.6 Mencari Derajat Keanggotaan Jarak

```
1   void distance_membership_degree(float jarak, float distance_max)
2      {
3
4           float batas_kiri_dekat = 25;
```

```
5           float batas_kanan_dekat = 40;
6           float batas_kiri_jauh = 35;
7           float batas_kanan_jauh = distance_max;
8
9           //membership dekat
10          if (jarak <= batas_kiri_dekat)
11          {
12              jar_dekat = 1;
13          }
14          if (jarak >= batas_kiri_dekat && jarak <= batas_kanan_dekat)
15          {
16              jar_dekat = ((batas_kanan_dekat - jarak) /
    (batas_kanan_dekat - batas_kiri_dekat));
17          }
18          if (jarak >= batas_kanan_dekat)
19          {
20              jar_dekat = 0;
21          }
22
23              //membership jauh
24          if (jarak <= batas_kiri_jauh)
25          {
26              jar_jauh = 0;
27          }
28          if (jarak >= batas_kiri_jauh && jarak <= batas_kanan_jauh)
29          {
30              jar_jauh = ((jarak - batas_kiri_jauh) / (batas_kanan_jauh
    - batas_kiri_jauh));
31          }
32          if (jarak >= batas_kanan_jauh)
33          {
34              jar_jauh = 1;
35          }
36      }
```

Berdasarkan Kode Sumber 5., baris 4-7 adalah perintah untuk mendeklarasikan dan menginisialisasikan variabel yang digunakan sebagai titik *fuzzy*. Selanjutnya pada baris 9-21 adalah perintah untuk mencari derajat keanggotaan kategori DEKAT berdasarkan Persamaan 4.3.1, sedangkan pada baris 23-35 adalah perintah untuk mencari derajat keanggotaan kategori LAMA berdasarkan Persamaan 4.3.2.

Kode Sumber 5.7 Proses Inferensi

```
1   void inference()
2       {
3           //Rule 1
4           a[0] = Mathf.Min(pos_depan, jar_jauh, offroad_lama);
5           o1[0] = (o1bataskanan_hard - ((o1bataskanan_hard -
    o1bataskiri_hard) * a[0]));
6           o2[0] = (o2bataskanan_hard - ((o2bataskanan_hard -
    o2bataskiri_hard) * a[0]));
7           o3[0] = (o3bataskanan_hard - ((o3bataskanan_hard -
    o3bataskiri_hard) * a[0]));
8           o4[0] = (o4bataskanan_hard - ((o4bataskanan_hard -
    o4bataskiri_hard) * a[0]));
9
10          //Rule 2
```

```
11          a[1] = Mathf.Min(pos_depan, jar_jauh, offroad_cepat);
12          o1[1] = (o1bataskanan_hard - ((o1bataskanan_hard -
    o1bataskiri_hard) * a[1]));
13          o2[1] = (o2bataskanan_hard - ((o2bataskanan_hard -
    o2bataskiri_hard) * a[1]));
14          o3[1] = (o3bataskanan_hard - ((o3bataskanan_hard -
    o3bataskiri_hard) * a[1]));
15          o4[1] = (o4bataskanan_hard - ((o4bataskanan_hard -
    o4bataskiri_hard) * a[1]));
16
17          //Rule 3
18          a[2] = Mathf.Min(pos_depan, jar_dekat, offroad_lama);
19          o1[2] = (o1bataskanan_hard - ((o1bataskanan_hard -
    o1bataskiri_hard) * a[2]));
20          o2[2] = (o2bataskanan_hard - ((o2bataskanan_hard -
    o2bataskiri_hard) * a[2]));
21          o3[2] = (o3bataskanan_hard - ((o3bataskanan_hard -
    o3bataskiri_hard) * a[2]));
22          o4[2] = (o4bataskanan_hard - ((o4bataskanan_hard -
23  o4bataskiri_hard) * a[2]));

24          //Rule 4
25          a[3] = Mathf.Min(pos_depan, jar_dekat, offroad_cepat);
26          o1[3] = (o1bataskanan_hard - ((o1bataskanan_hard -
27  o1bataskiri_hard) * a[3]));
            o2[3] = (o2bataskanan_hard - ((o2bataskanan_hard -
28  o2bataskiri_hard) * a[3]));
            o3[3] = (o3bataskanan_hard - ((o3bataskanan_hard -
29  o3bataskiri_hard) * a[3]));
            o4[3] = (o4bataskanan_hard - ((o4bataskanan_hard -
30  o4bataskiri_hard) * a[3]));

31          //Rule 5
32          a[4] = Mathf.Min(pos_medium, jar_dekat, offroad_cepat);
33          o1[4] = (o1bataskanan_medium - ((o1bataskanan_medium -
34  o1bataskiri_medium) * a[4]));
            o2[4] = (o2bataskanan_medium - ((o2bataskanan_medium -
35  o2bataskiri_medium) * a[4]));
            o3[4] = (o3bataskanan_medium - ((o3bataskanan_medium -
36  o3bataskiri_medium) * a[4]));
            o4[4] = (o4bataskanan_medium - ((o4bataskanan_medium -
37  o4bataskiri_medium) * a[4]));

38          //Rule 6
39          a[5] = Mathf.Min(pos_medium, jar_dekat, offroad_lama);
40          o1[5] = (o1bataskanan_medium - ((o1bataskanan_medium -
41  o1bataskiri_medium) * a[5]));
            o2[5] = (o2bataskanan_medium - ((o2bataskanan_medium -
42  o2bataskiri_medium) * a[5]));
            o3[5] = (o3bataskanan_medium - ((o3bataskanan_medium -
43  o3bataskiri_medium) * a[5]));
            o4[5] = (o4bataskanan_medium - ((o4bataskanan_medium -
44  o4bataskiri_medium) * a[5]));

45          //Rule 7
46              a[6] = Mathf.Min(pos_medium, jar_jauh, offroad_cepat);
47              o1[6] = (o1bataskanan_medium - ((o1bataskanan_medium -
48  o1bataskiri_medium) * a[6]));
                o2[6] = (o2bataskanan_medium - ((o2bataskanan_medium -
49  o2bataskiri_medium) * a[6]));
```

```
            o3[6] = (o3bataskanan_medium - ((o3bataskanan_medium -
50  o3bataskiri_medium) * a[6]));
            o4[6] = (o4bataskanan_medium - ((o4bataskanan_medium -
51  o4bataskiri_medium) * a[6]));

52          //Rule 8
53          a[7] = Mathf.Min(pos_medium, jar_jauh, offroad_lama);
54          o1[7] = (o1bataskanan_medium - ((o1bataskanan_medium -
    o1bataskiri_medium) * a[7]));
55          o2[7] = (o2bataskanan_medium - ((o2bataskanan_medium -
    o2bataskiri_medium) * a[7]));
56          o3[7] = (o3bataskanan_medium - ((o3bataskanan_medium -
    o3bataskiri_medium) * a[7]));
57          o4[7] = (o4bataskanan_medium - ((o4bataskanan_medium -
    o4bataskiri_medium) * a[7]));
58
59          //Rule 9
60          a[8] = Mathf.Min(pos_belakang, jar_dekat, offroad_lama);
61          o1[8] = (o1bataskiri_easy + ((o1bataskanan_easy -
62  o1bataskiri_easy) * a[8]));
            o2[8] = (o2bataskiri_easy + ((o2bataskanan_easy -
63  o2bataskiri_easy) * a[8]));
            o3[8] = (o3bataskiri_easy + ((o3bataskanan_easy -
64  o3bataskiri_easy) * a[8]));
            o4[8] = (o4bataskiri_easy + ((o4bataskanan_easy -
65  o4bataskiri_easy) * a[8]));

66          //Rule 10
67          a[9] = Mathf.Min(pos_belakang, jar_dekat, offroad_cepat);
68          o1[9] = (o1bataskiri_easy + ((o1bataskanan_easy -
69  o1bataskiri_easy) * a[9]));
            o2[9] = (o2bataskiri_easy + ((o2bataskanan_easy -
70  o2bataskiri_easy) * a[9]));
            o3[9] = (o3bataskiri_easy + ((o3bataskanan_easy -
71  o3bataskiri_easy) * a[9]));
            o4[9] = (o4bataskiri_easy + ((o4bataskanan_easy -
72  o4bataskiri_easy) * a[9]));

73           //Rule 11
74           a[10] = Mathf.Min(pos_belakang, jar_jauh, offroad_lama);
75           o1[10] = (o1bataskiri_easy + ((o1bataskanan_easy -
76  o1bataskiri_easy) * a[10]));
            o2[10] = (o2bataskiri_easy + ((o2bataskanan_easy -
77  o2bataskiri_easy) * a[10]));
            o3[10] = (o3bataskiri_easy + ((o3bataskanan_easy -
78  o3bataskiri_easy) * a[10]));
            o4[10] = (o4bataskiri_easy + ((o4bataskanan_easy -
79  o4bataskiri_easy) * a[10]));

80          //Rule 12
81          a[11] = Mathf.Min(pos_belakang, jar_jauh, offroad_cepat);
82          o1[11] = (o1bataskiri_easy + ((o1bataskanan_easy -
83  o1bataskiri_easy) * a[11]));
            o2[11] = (o2bataskiri_easy + ((o2bataskanan_easy -
84  o2bataskiri_easy) * a[11]));
            o3[11] = (o3bataskiri_easy + ((o3bataskanan_easy -
    o3bataskiri_easy) * a[11]));
85          o4[11] = (o4bataskiri_easy + ((o4bataskanan_easy -
    o4bataskiri_easy) * a[11]));
86      }
```

Berdasarkan Kode Sumber 5., dicari α-predikat dengan menggunakan fungsi Mathf.Min, kemudian nilai z untuk setiap parameter keluaran akan dicari nilai z (o1, o2, o3, o4) menggunakan Persamaan 2.7, nilai α-predikat dan nilai z dicari untuk setiap *rule* yang ada.

Pada baris 3-9 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 1. Baris 11-16 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 2. Baris 18-23 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 3. Baris 25-30 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 4. Baris 32-37 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 5. Baris 39-44 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 6. Baris 46-51 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 7. Baris 53-58 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 8. Baris 60-65 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 9. Baris 67-72 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 10. Baris 74-79 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 11. Baris 81-86 adalah perintah untuk mencari nilai α-predikat dan nilai z untuk *rule* 12. Selanjutnya masuk ke dalam fungsi Defuzzyfication, untuk kode sumbernya dipaparkan pada Kode Sumber 5..

Kode Sumber 5.8 Proses Defuzzyfication

```
1   void defuzzyfication()
2       {
3           sumAZ1 = 0;
4           sumAZ2 = 0;
5           sumAZ3 = 0;
6           sumAZ4 = 0;
7           sumA = 0;
8
9           for (int i = 0; i < 12; i++)
10          {
11              sumAZ1 += a[i] * o1[i]; // Output avoidance sensitivity
12              sumAZ2 += a[i] * o2[i]; // Output steer sensitivity
13              sumAZ3 += a[i] * o3[i]; // Output max wander distance
14              sumAZ4 += a[i] * o4[i]; // Output wander rate
15              sumA += a[i]; //menghitung
16          }
17          //menghitung nilai Z
18          O1 = sumAZ1 / sumA;
19          O2 = sumAZ2 / sumA;
20          O3 = sumAZ3 / sumA;
21          O4 = sumAZ4 / sumA;
22          cautionAngle = O1;
23          steerSensitivity = O2;
24          maxWanderDistance = O3;
25          wanderRate = O4;
26          offRoadDuration = 0;
27      }
```

Berdasarkan Kode Sumber 5., pada baris 3-7, sumAZ dikembalikan menjadi 0 agar nilai tidak terganggu seiring perulangan implementasi DDA *Fuzzy* dijalankan. Pada baris 9-16 dilakukan perhitungan untuk menghitung hasil

penjumlahan seluruh α-predikat dikalikan dengan nilai z (sumAZ1, sumAZ2, sumAZ3, sumAZ4) untuk setiap *rule* dan juga mencari nilai sumA (total α-predikat seluruh *rule).* Selanjutnya pada baris 19-22 akan dilakukan perhitungan untuk mendapatkan nilai Z(O1, O2, O3, O4) dengan membagikan sumAZ untuk semua parameter keluaran dengan sumA.

Kemudian pada baris 23-26, nilai Z yang telah diperoleh sebelumnya dimasukkan ke dalam parameter cautionAngle, steerSensitivity, maxWanderDistance, dan WanderRate untuk mengubah nilai parameter masukan yang akan digunakan dalam perulangan berikutnya. Kemudian pada baris 27 offroadDuration dikembalikan menjadi 0 agar tidak menganggu proses implementasi DDA pada perulangan berikutnya.

BAB 6 PENGUJIAN DAN ANALISIS

6.1 Pengujian Parameter *Fuzzy*

Dalam pengujian parameter *fuzzy*, akan dilakukan pengujian yang bertujuan untuk mengetahui seberapa besar dampak yang diberikan oleh parameter yang digunakan terhadap DDA. Pengujian ini mengacu pada DE yang terdapat proses pengujian *Baseline* didalamnya, yang mana setiap *bot* statis dengan perilaku yang berbeda akan ditandingkan. Pengujian dilakukan dengan cara mengadukan tiga *bot* yang memiliki perilaku berbeda yang terdiri dari *bot easy, bot medium,* dan *bot hard.* Pengujian dilakukan sebanyak tiga kali pengujian. Adapun tahap pengujian yang dilakukan adalah sebagai berikut:

1. Menyetarakan kemampuan mobil dari setiap *bot*, hal ini dilakukan untuk mengimbangkan potensi kemampuan tiap mobil yang digunakan *bot* guna memperoleh hasil pengujian yang lebih spesifik terhadap parameter uji.
2. Mengubah nilai setiap parameter uji sesuai dengan perilaku *bot* tersebut. Adapun pemaparan pengaturannya yang digunakan oleh peneliti dirincikan pada Tabel 6.1.

Tabel 6.1 Rincian Pengaturan Perilaku *Bot*

No	Parameter uji	*Easy*	*Medium*	*Hard*
1	Steer Sensitivity	0,075	0,01	0,03
2	Max Wander Distance	25	20	3
3	Wander Rate	0,2	0,15	0,03
4	Caution Angle	20	30	45

Nilai-nilai dari Tabel 6.1 diperoleh dari proses *tweaking* yang dilakukan berulang kali sehingga memperoleh nilai yang mampu memberikan dampak perbedaan kemampuan yang terlihat berbeda untuk setiap *bot* dan memberikan hasil yang optimal pada DDA.

3. Mengatur posisi awal dengan *bot easy* berada pada posisi paling depan, *bot medium* pada posisi tengah, dan *bot hard* berada pada posisi paling belakang. Hal ini dilakukan untuk mendapatkan hasil yang lebih meyakinkan.
4. Memulai pengujian dengan mode balapan *circuit* yang terdiri dari tiga *lap*, pengujian diulangi sebanyak tiga kali untuk memperoleh hasil yang lebih akurat dan meyakinkan.
5. Mengobservasi dan mencatat hasil pengujian, kemudian mengambil kesimpulan berdasarkan hasil pengujian yang dilakukan.

Berikut adalah pemaparan hasil pengujian yang dilakukan berdasarkan langkah-langkah di atas :

1. Pengujian pertama

Tabel 6.2 Hasil Pengujian Pertama Pengujian Parameter

No	Pembalap	Lap 1	Lap 2	Lap 3	Total Lap	Posisi
1	*Bot easy*	01:14:13	01:03:23	01:16:85	03:35:01	3
2	*Bot medium*	01:05:02	00:57:70	00:57:43	03:20:55	2
3	*Bot hard*	01:01:69	00:53:54	00:52:71	02:49:14	1

2. Pengujian kedua

Tabel 6.3 Hasil Pengujian Kedua Pengujian Parameter

No	Pembalap	Lap 1	Lap 2	Lap 3	Total Lap	Posisi
1	*Bot easy*	01:12:89	01:26:39	01:23:50	04:02:79	3
2	*Bot medium*	01:08:15	00:58:19	00:58:19	03:03:16	2
3	*Bot hard*	01:01:42	00:53:75	00:51:74	02:46:92	1

3. Pengujian ketiga

Tabel 6.4 Hasil Pengujian Ketiga Pengujian Parameter

No	Pembalap	Lap 1	Lap 2	Lap 3	Total Lap	Posisi
1	*Bot easy*	01:14:49	01:03:51	01:15:26	03:33:53	3
2	*Bot medium*	01:20:71	00:56:29	01:06:98	03:24:00	2
3	*Bot hard*	01:05:12	00:52:25	00:51:60	02:49:80	1

6.2 Pengujian DDA

Dalam pengujian DDA, akan dilakukan pengujian yang bertujuan untuk mengevaluasi apakah hasil keluaran yang diberikan oleh DDA menggunakan *fuzzy* yang telah diterapkan mampu memberikan hasil yang sesuai dengan tujuan penelitian, yaitu mengimbangkan kemampuan pemain dengan kemampuan *bot*. Pengujian ini mengacu pada DE yang mana terdapat pengujian *adaptive bot*

melawan *bot* statis. Pengujian dibagi menjadi dua, yaitu pengujian *bot* statis melawan *bot* DDA dan pengujian pemain melawan *bot* DDA dan *bot* statis.

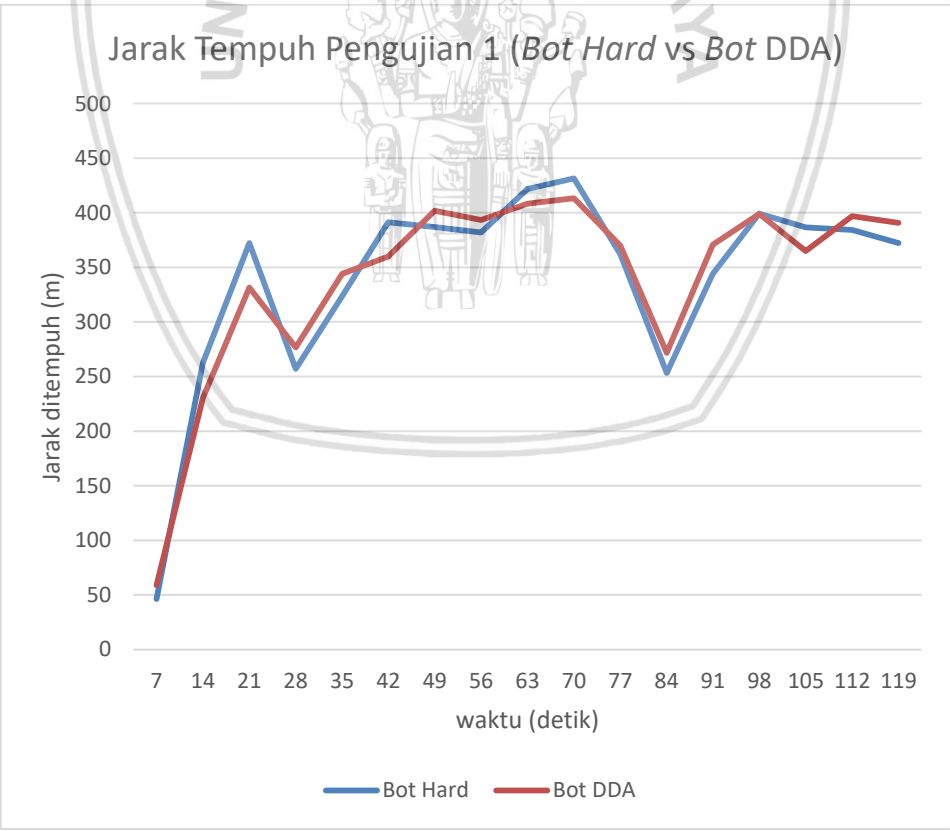
6.2.1 Pengujian *bot* statis melawan *bot* DDA

Pengujian *bot* statis melawan *bot* DDA dilakukan dengan cara menandingkan *bot* yang perilakunya statis dan *bot* yang menggunakan DDA dengan metode *fuzzy*. Kemampuan mobil untuk *bot* statis dan *bot* DDA disetarakan. Perilaku *bot* statis telah diatur sesuai dengan perilaku untuk *bot easy, medium*, dan *hard* sebelum pengujian dilakukan. Nilai-nilai parameter *caution angle, steer sensitivity, max wander distance,* dan *wander rate* disesuaikan seperti yang tertera pada Tabel 6.1. Pengujian dilakukan sebanyak tiga kali untuk setiap tingkat kemampuan (*easy*, *medium*, *hard*).

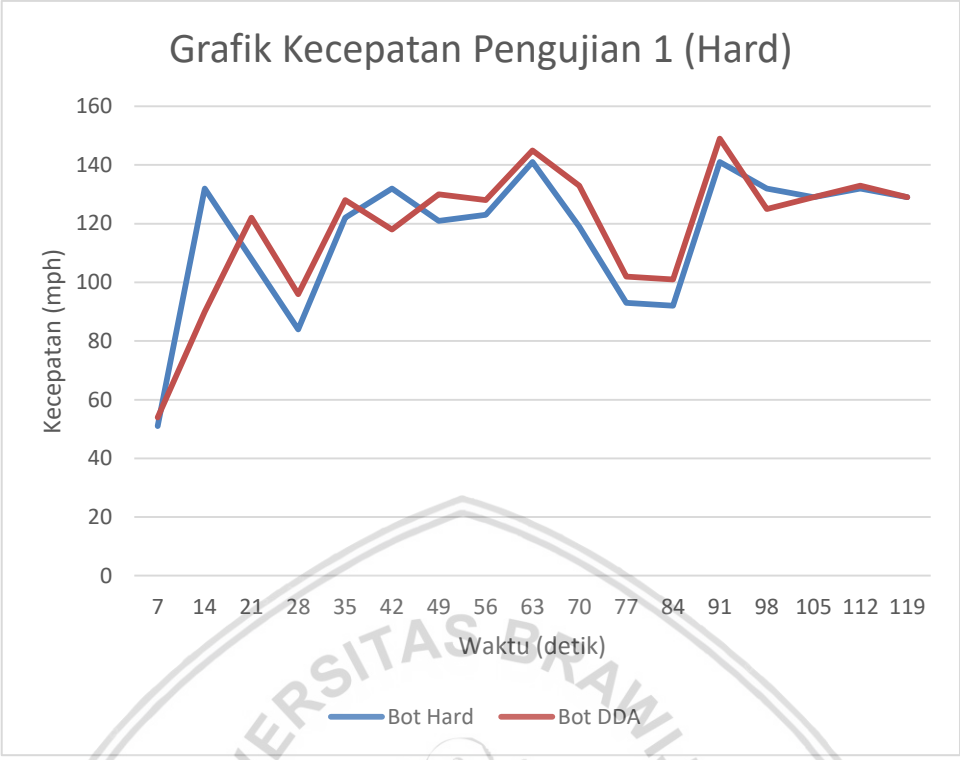
Data yang dipaparkan adalah data jarak yang ditempuh oleh *bot* berdasarkan jarak sebenarnya dari lintasan dalam satuan meter, dan kecepatan yang diperoleh dalam satuan mph (*miles per hour*). Data diambil tiap perulangan DDA dipanggil yaitu setiap tujuh detik. Kedua satuan meter dan mph merupakan satuan langsung yang digunakan dalam RGSK, segala perhitungan dan besaran satuannya langsung diperoleh dari RGSK.

Berikut merupakan paparan hasil pengujian untuk setiap perilaku:

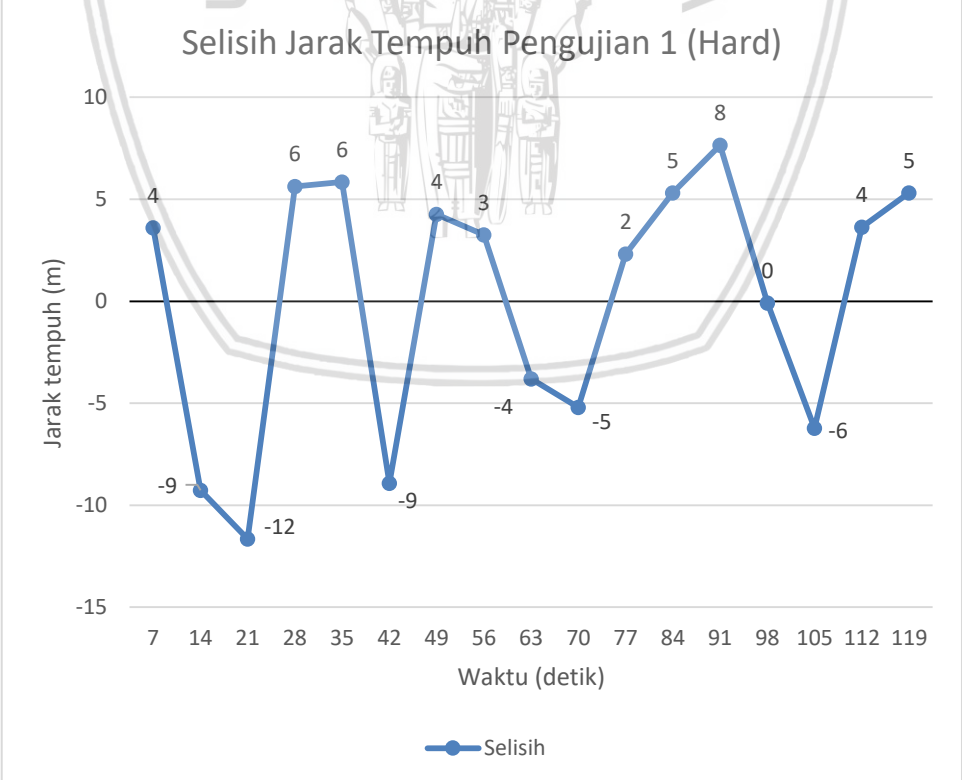
1. Pengujian pertama (*hard*)



Gambar 6.1 Grafik Jarak Tempuh Pengujian 1 *(Hard)*

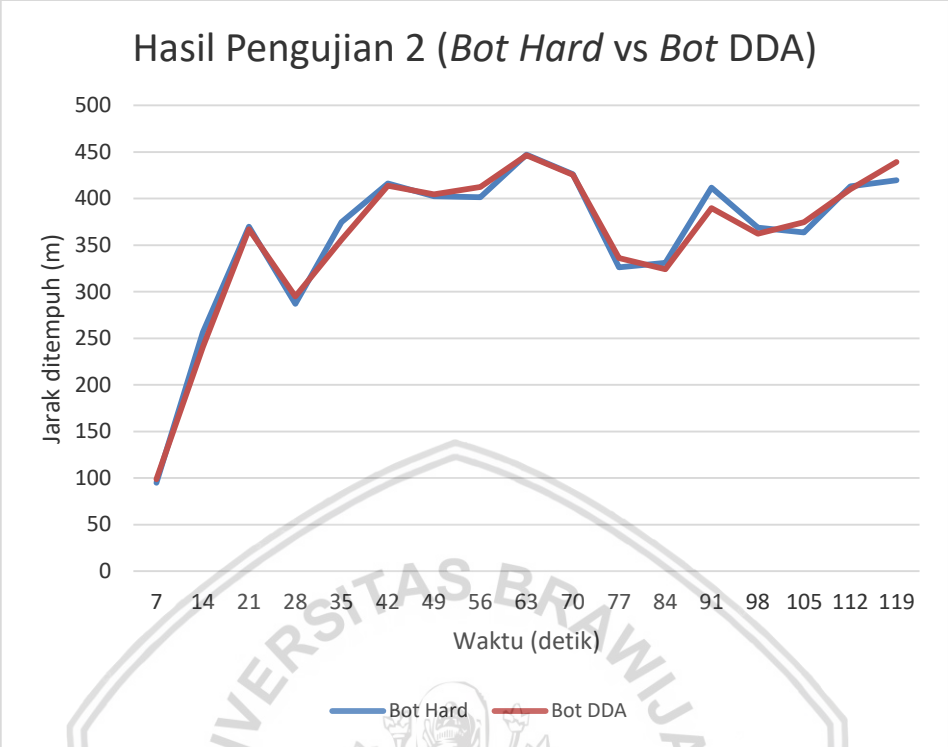


Gambar 6.2 Grafik Kecepatan Pengujian 1 *(Hard)*



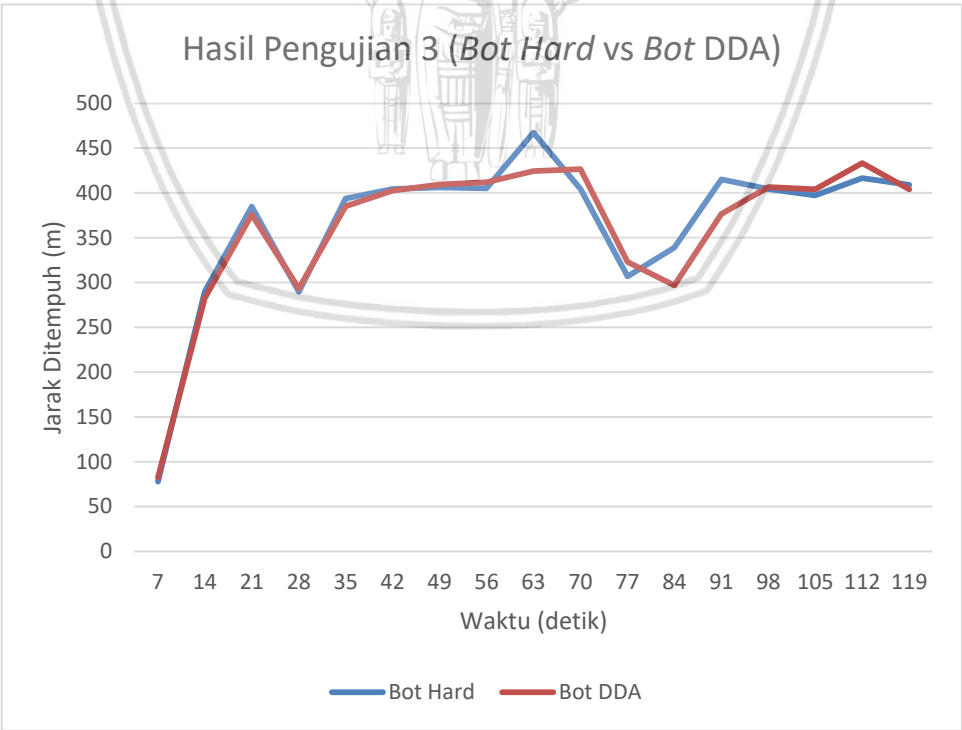
Gambar 6.3 Selisih Kumulatif Total Tempuh Pengujian 1 (Hard)

2. Pengujian kedua (*hard*)



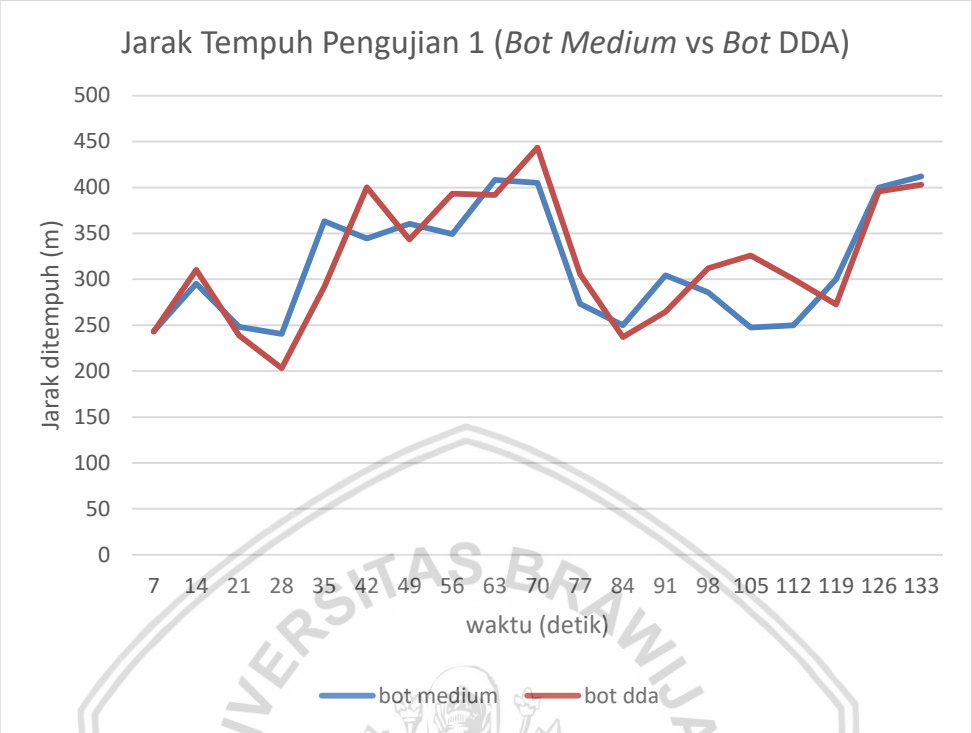
Gambar 6.4 Hasil Pengujian Kedua *Bot* Statis Melawan *Bot* DDA *(Hard)*

3. Pengujian ketiga (*hard*)

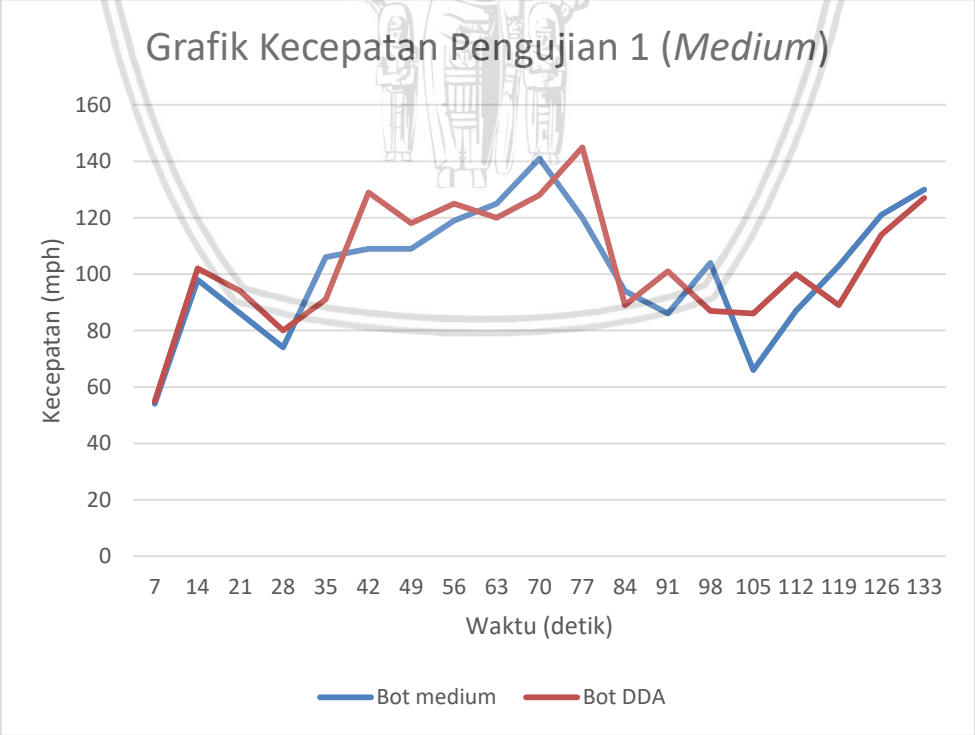


Gambar 6.5 Hasil Pengujian Ketiga *Bot* Statis Melawan *Bot* DDA *(Hard)*

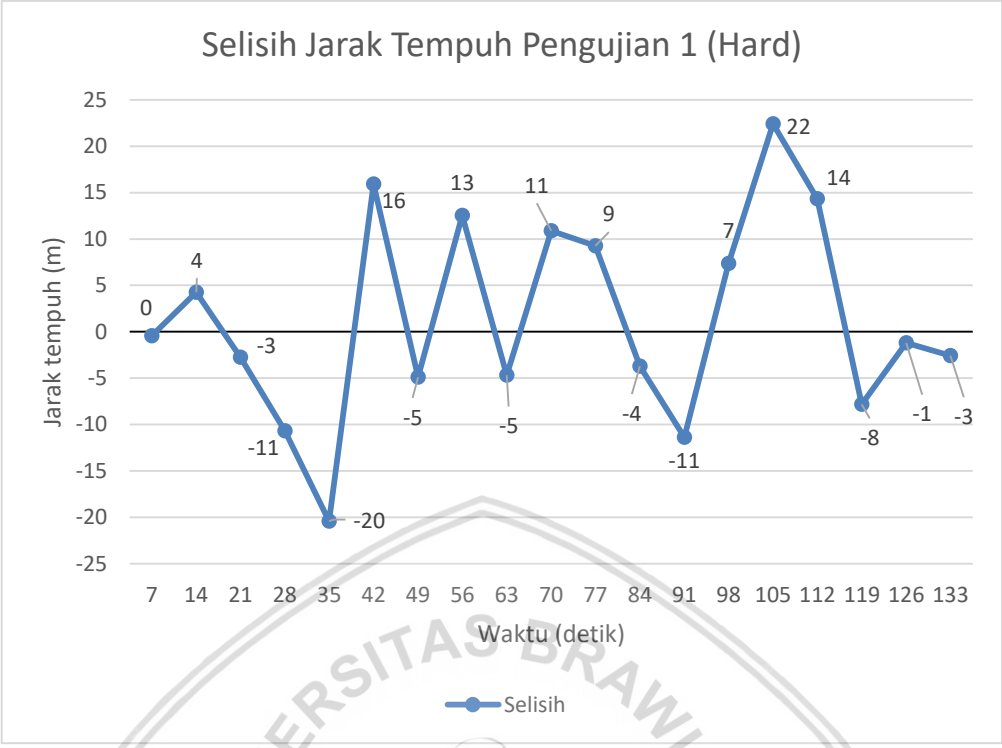
4. Pengujian pertama (*medium*)



Gambar 6.6 Grafik Jarak Tempuh Pengujian *Bot* DDA *(Medium)*

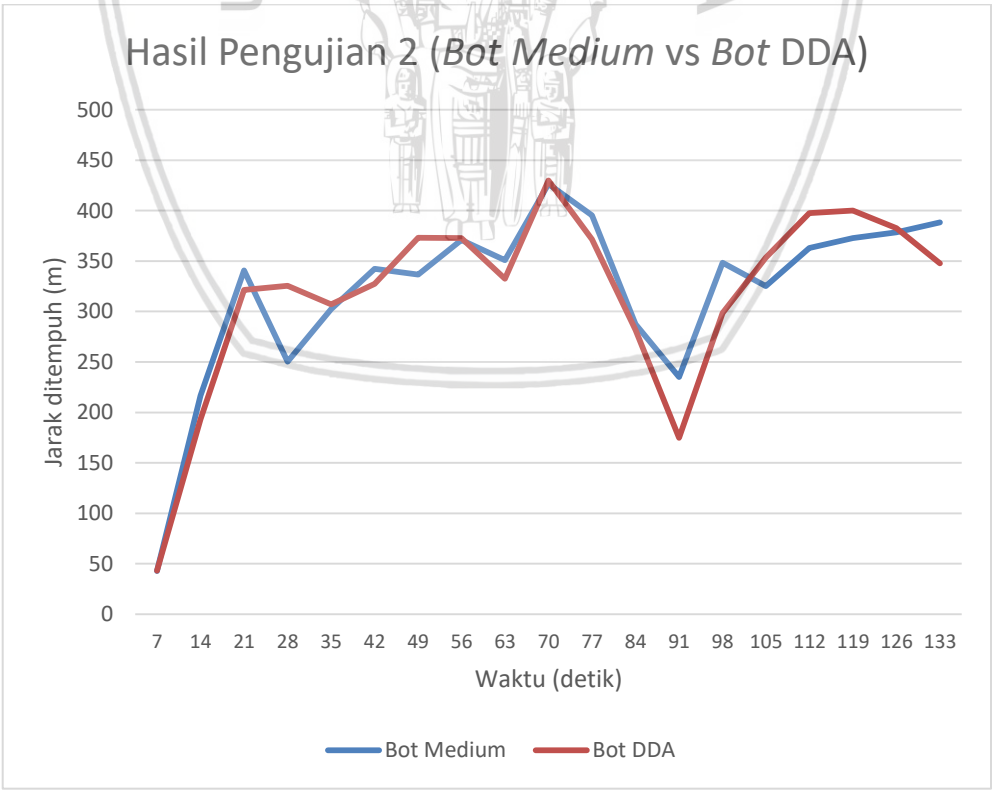


Gambar 6.7 Grafik Kecepatan Pengujian 1 (Medium)



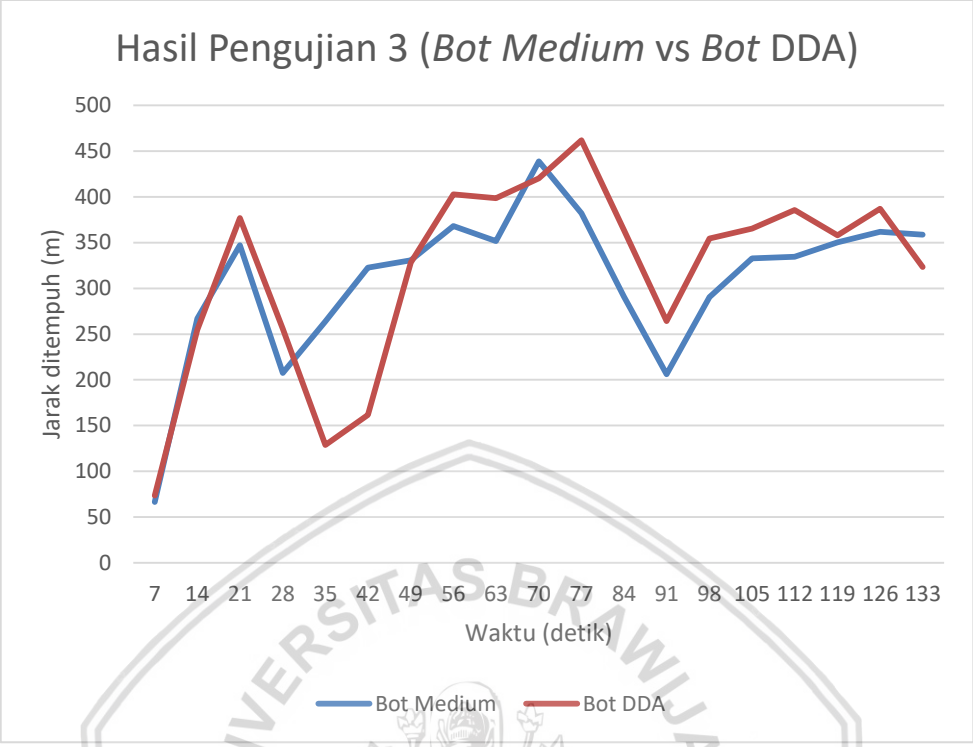
Gambar 6.8 Selisih Kumulatif Total Tempuh Pengujian 1 (Medium)

5. Pengujian kedua (*medium*)



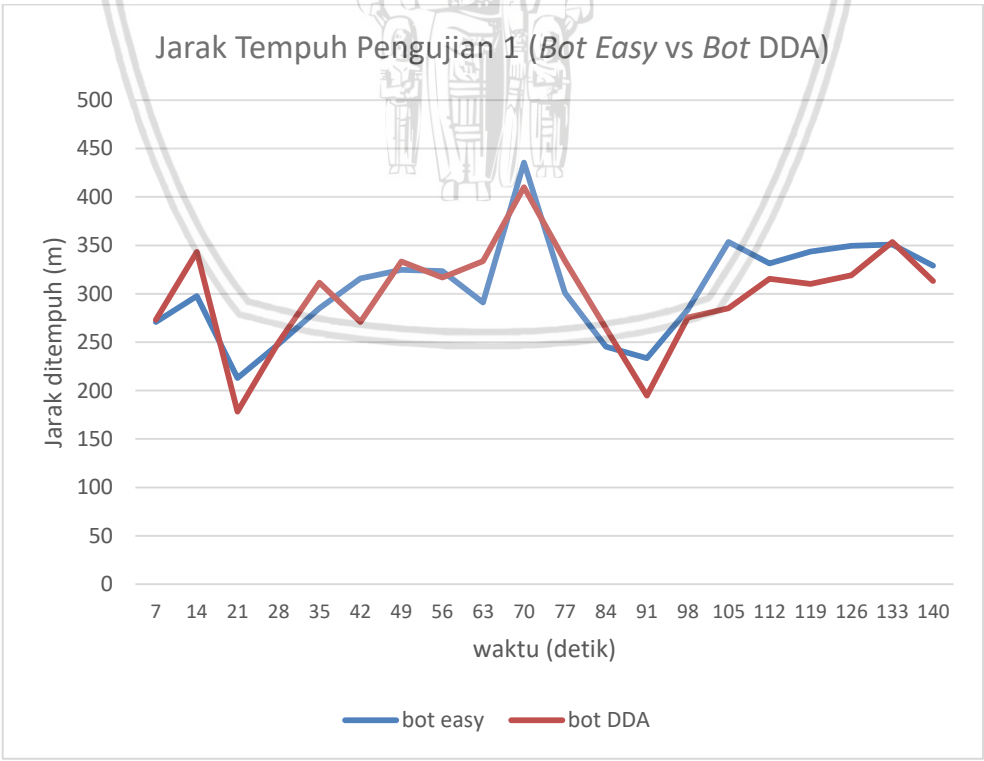
Gambar 6.9 Hasil Pengujian Kedua *Bot* Statis Melawan *Bot* DDA *(Medium)*

6. Pengujian ketiga (*medium*)

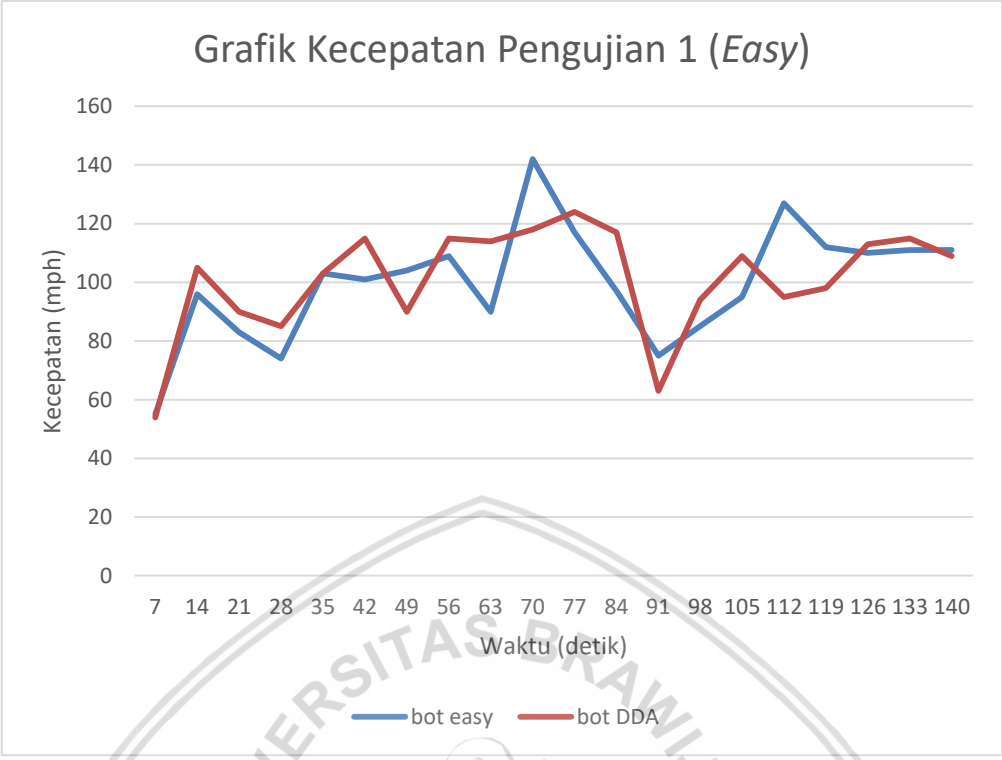


Gambar 6.10 Hasil Pengujian Ketiga *Bot* Statis Melawan *Bot* DDA *(Medium)*

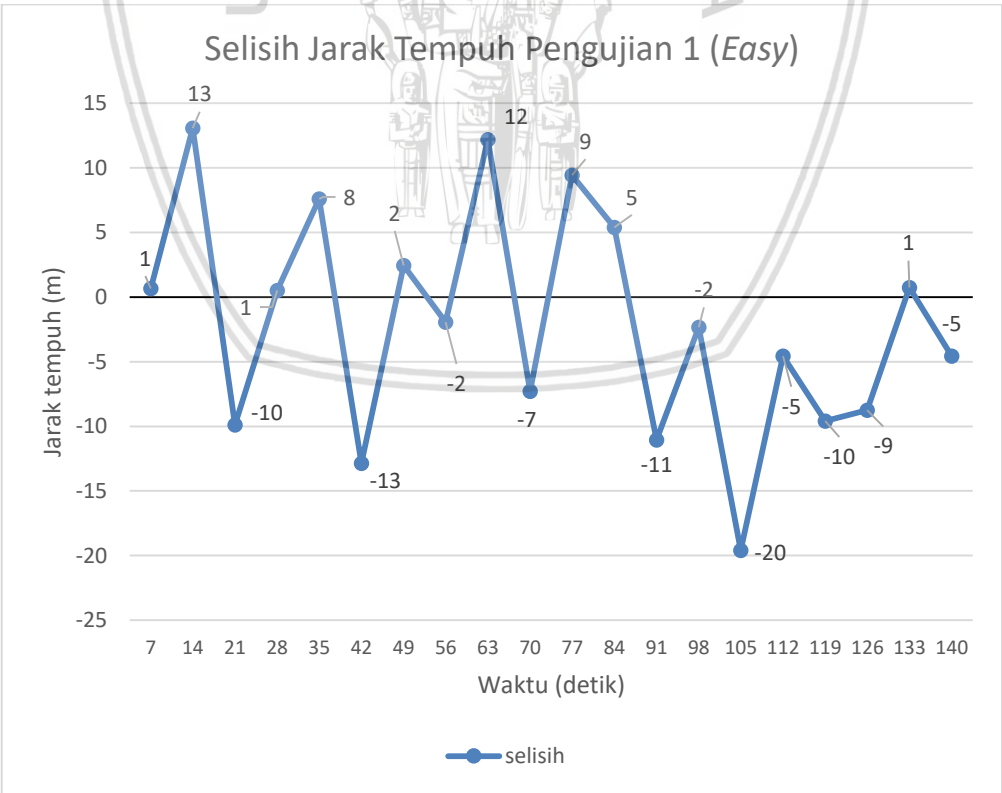
7. Pengujian pertama (*easy*)



Gambar 6.11 Grafik Jarak Tempuh Pengujian 1 *(Easy)*

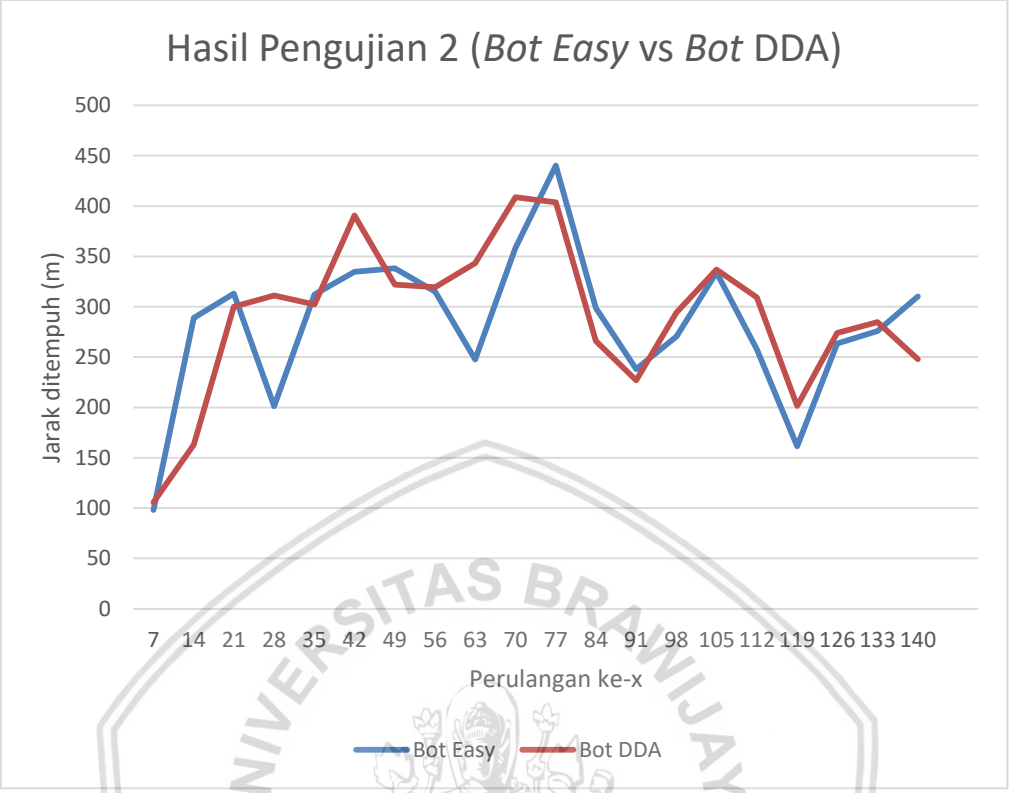


Gambar 6.12 Grafik Kecepatan Pengujian 1 (*Easy*)



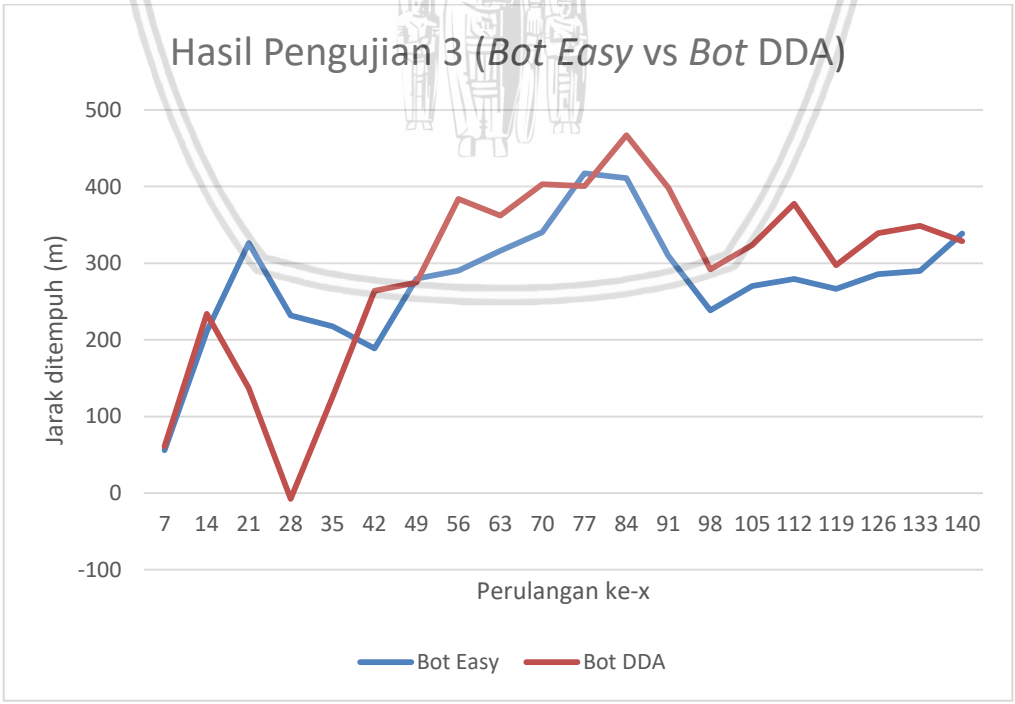
Gambar 6.13 Selisih Kumulatif Total Tempuh Pengujian 1 (*Easy*)

8. Pengujian kedua (*easy*)



Gambar 6.14 Hasil Pengujian Kedua *Bot* Statis Melawan *Bot* DDA *(Easy)*

9. Pengujian ketiga (*easy*)



Gambar 6.15 Hasil Pengujian Ketiga *Bot* Statis Melawan *Bot* DDA *(Easy)*

6.2.2 Pengujian pemain melawan *bot* DDA dan *bot* statis

Pengujian dilakukan dengan cara mengujikan *game racing* yang telah ditambahkan DDA dengan metode *fuzzy* dan tanpa menggunakan DDA dengan metode *fuzzy* kepada penguji secara langsung. Penguji terdiri dari satu pemain yang berpengalaman dengan kemampuan yang baik, dan satu pemain pemula dengan kemampuan yang rendah.

Berikut merupakan data-data penguji yang terkait dengan *racing game* sebagai bukti untuk membantu menyatakan kemampuan yang mereka miliki :

1. Penguji Kesatu (Handal) :
   a. 15 tahun pengalaman memainkan *racing game*.
   b. Beberapa *racing game* yang pernah dimainkan adalah :

Tabel 6.5 Daftar *Racing Game* yang Pernah dimainkan Penguji Handal

No	Judul *Game*	Status
1	Need for Speed Most Wanted	Tamat
2	Need for Speed Underground	Tamat
3	Need for Speed Underground 2	Tamat
4	Gran Turismo	Belum Tamat
5	Need for Speed Carbon	Tamat
6	Grid	Tamat
7	Grid 2	Belum Tamat
8	Shift	Belum Tamat
9	Midnight Club (Tamat)	Tamat
10	The Crew	Belum Tamat
12	F1 2010	Belum Tamat
13	F1 2012	Belum Tamat
14	Need for Speed Hot Pursuit	Belum Tamat
16	Need for Speed Undercover	Tamat
17	Need for Speed ProStreet	Tamat
18	Nascar Thunder	Belum Tamat
19	Nascar Rumble	Belum Tamat
20	Need for Speed The Run	Belum Tamat
21	Burnout	Tamat
22	Burnout 2	Tamat

2. Penguji Kedua (Pemula) :
   a. Mulai memainkan *racing game* sejak SMP (Terhitung 10 tahun)
   b. Beberapa *racing game* yang pernah dimainkan adalah :

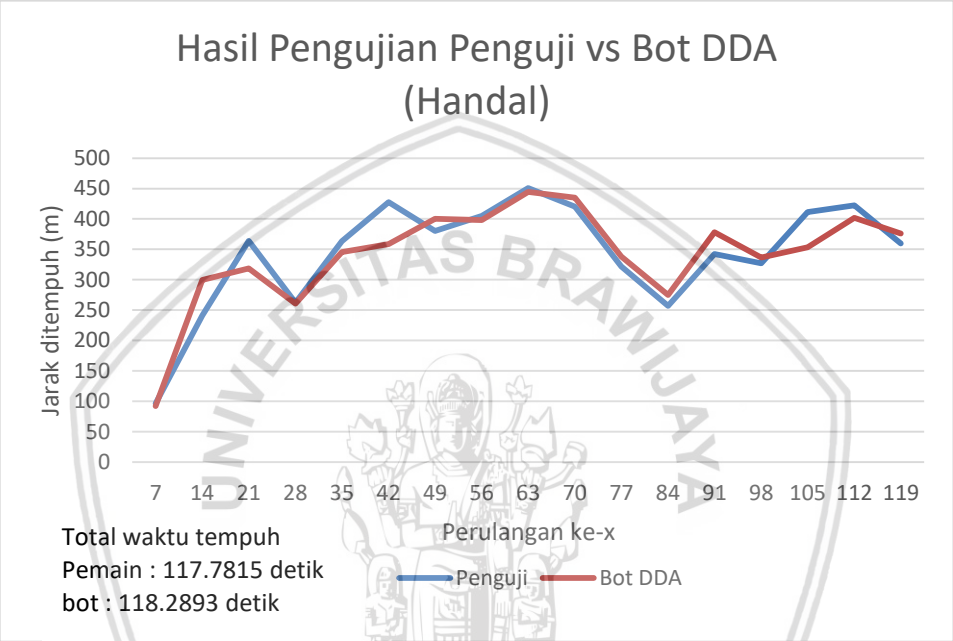
Tabel 6.6 Daftar *Racing Game* yang Pernah dimainkan Penguji Pemula

No	Judul *Game*	Status
1	Need for Speed Most Wanted	Belum tamat
2	Need for Speed Underground 2	Belum tamat
3	Need for Speed Carbon	Belum tamat

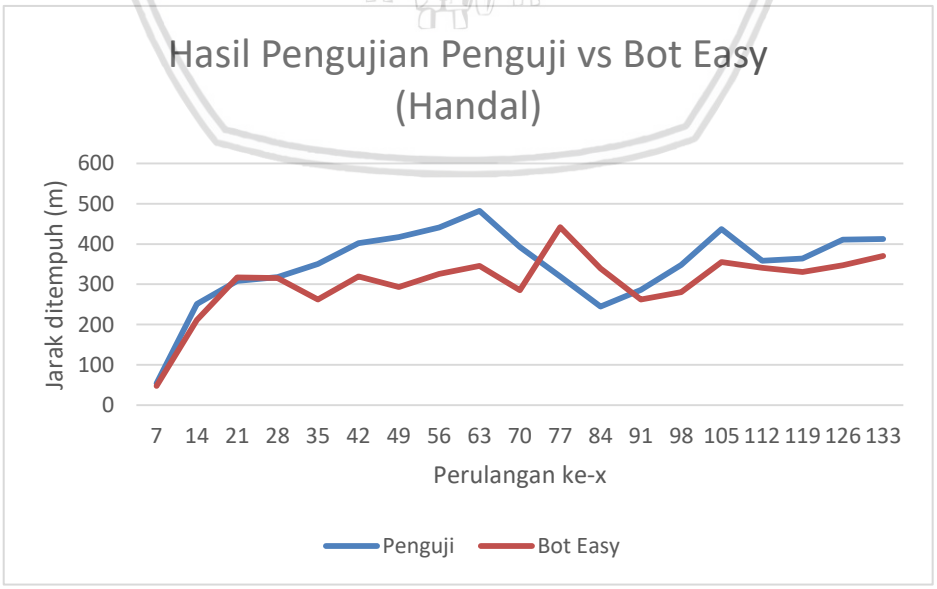
Berdasarkan data dari Tabel 6.5, dapat dilihat pengalaman yang dimiliki oleh penguji kesatu sangat banyak, dari pengalaman-pengalaman tersebut penguji ini dapat dikatakan sebagai penguji handal. Berdasarkan data dari Tabel 6.6, penguji kedua tidak memiliki pengalaman yang banyak dalam memainkan *racing game* dan belum pernah menyelesaikan satupun *racing game*, sehingga dapat dikatakan penguji kedua adalah penguji pemula.

Berikut merupakan paparan hasil pengujian yang telah diuji kepada para penguji :

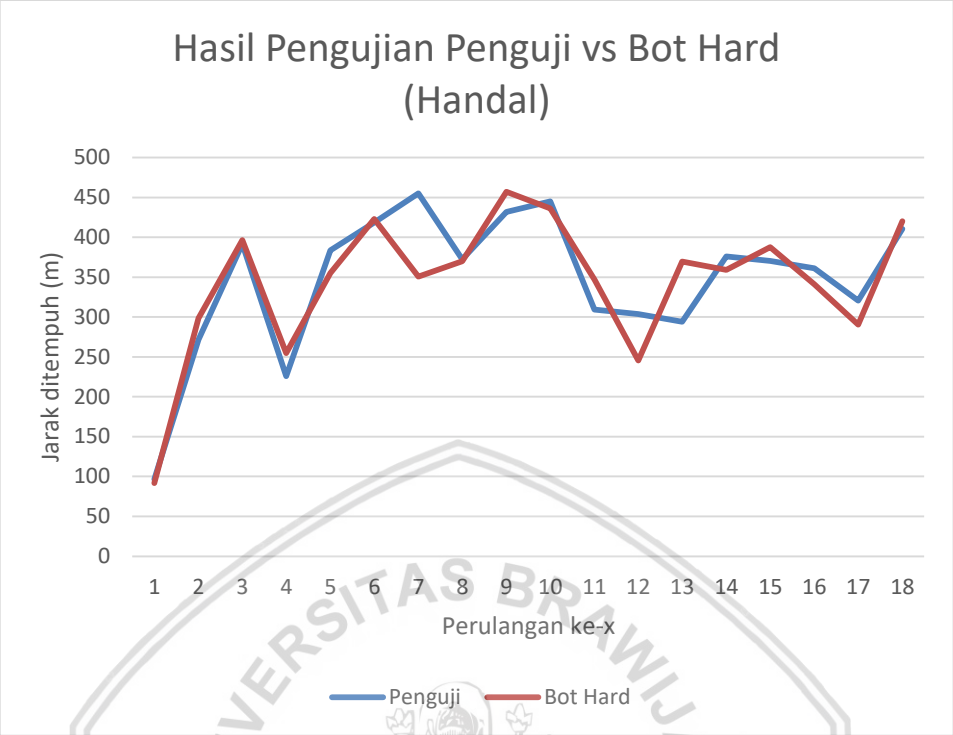
1. Penguji Handal



Gambar 6.16 Hasil Pengujian Penguji Handal Melawan *Bot* DDA

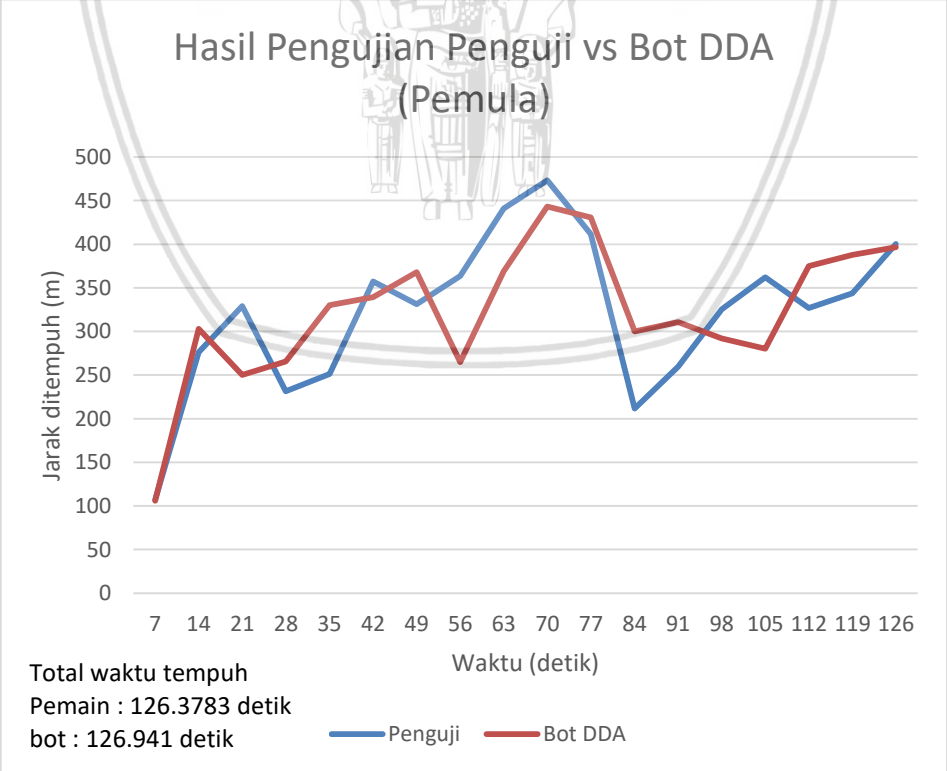


Gambar 6.17 Hasil Pengujian Penguji Handal Melawan *Bot Easy*

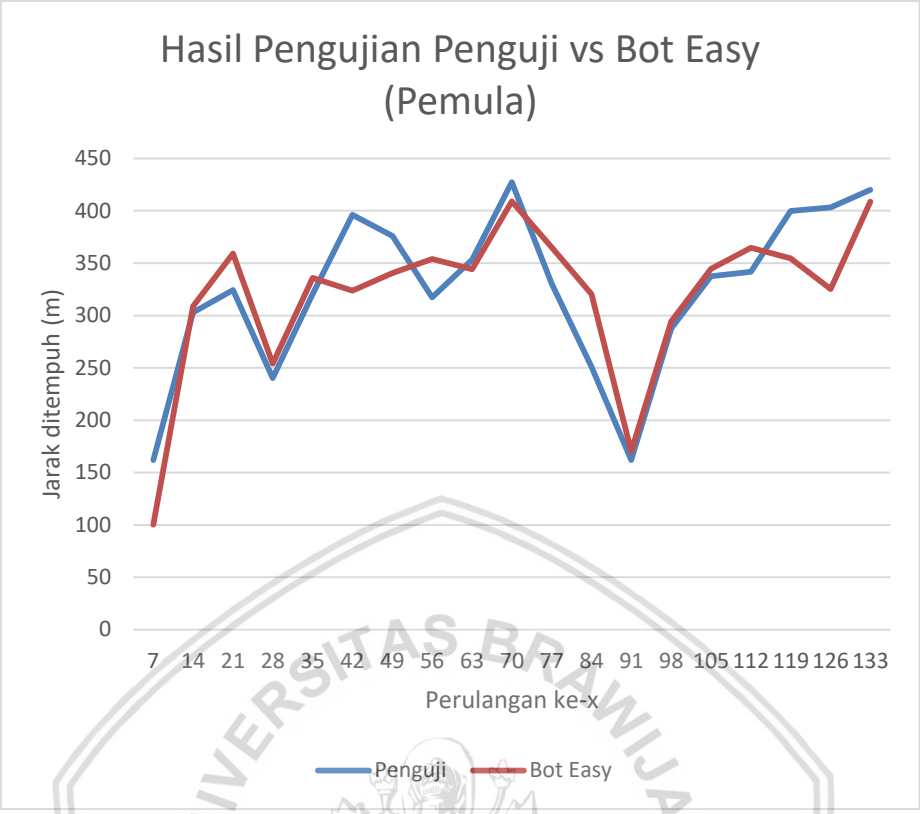


Gambar 6.18 Hasil Pengujian Penguji Handal Melawan *Bot Hard*

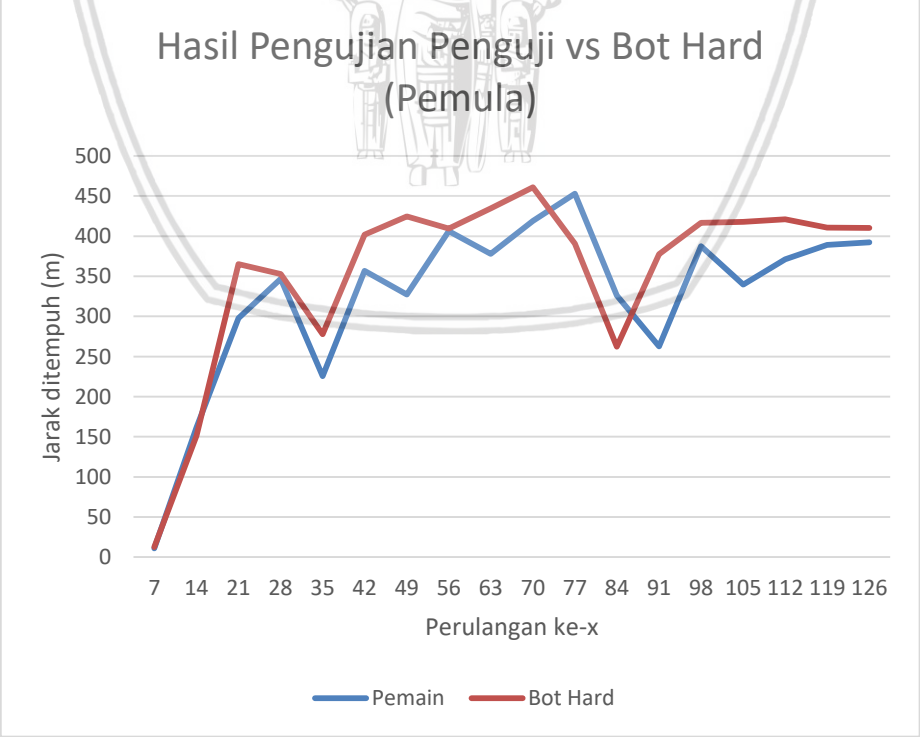
2. Penguji Pemula



Gambar 6.19 Hasil Pengujian Penguji Pemula Melawan *Bot* DDA



Gambar 6.20 Hasil Pengujian Penguji Pemula Melawan *Bot Easy*



Gambar 6.21 Hasil Pengujian Penguji Pemula Melawan *Bot Hard*

6.3 Analisis Pengujian

6.3.1 Analisis Pengujian Parameter Fuzzy

Berdasarkan hasilnya dari Tabel 6.2, Tabel 6.3, dan Tabel 6.4, terlihat jelas bahwa dari ketiga kali percobaan yang telah dilakukan *bot hard* selalu mengungguli *bot medium* dan *bot easy*. Begitu juga untuk *bot easy*, selalu berada pada posisi terakhir dengan perbedaan jarak yang signifikan. Hal ini menandakan bahwa penggunaan keempat parameter ini dapat memberikan perbedaan perilaku yang signifikan terhadap kemampuan *bot*.

6.3.2 Analisis Pengujian DDA

a. *Bot* DDA Melawan *Bot Hard*

Berdasarkan hasil dari Gambar 6.1 pada pengujian pertama perilaku *Hard*, terjadi beberapa kali perpotongan garis yang menunjukkan adanya kejar mengejar yang terjadi antara *bot* DDA dan *bot hard*. Pada waktu detik ke-35 hingga detik 49 terjadi dua kali perpotongan garis, yaitu pada detik ke-35 menuju detik ke-42, dan detik 42 menuju detik ke-49. Pada detik ke-42 *bot hard* memiliki jarak tempuh yang lebih tinggi dibandingkan dengan *bot* DDA, dikarenakan pada detik ke-35 terjadi perubahan nilai parameter keluaran seperti pada Tabel 6.7.

Tabel 6.7 Perubahan Parameter Pengujian 1 *Hard* Detik ke-28 dan Detik ke-35

Waktu (detik)	*Caution Angle*	*Steer Sensitivity*	*Max Wander Distance*	*Wander Rate*
28	40	0,008	1	0
35	25,63	0,004	18,061	0,19

Berdasarkan Tabel 6.7, nilai parameter pada detik ke-28 termasuk perilaku *Hard* karena nilai *caution angle* berada di antara 40 sampai 50, nilai *steer sensitivity* berada di antara 0,008 sampai 0,03, *max wander distance* berada di antara 1 sampai 15, dan *wander rate* bernilai di antara 0 sampai 0,16, sedangkan pada detik ke-35 masuk ke dalam kategori *Medium* karena nilai *caution angle* berada di antara 20 sampai 45, nilai *steer sensitivity* berada di antara 0,004 sampai 0,009, *max wander distance* berada di antara 14 sampai 20, dan nilai *wander rate* berada di antara 0,15 sampai 0,20, sesuai seperti rancangan parameter pada Bab 2. Karena terjadinya perubahan nilai parameter tadi, maka dapat dilihat pada Gambar 6.2 perulangan detik 35 menuju detik , terjadi penurunan kecepatan yang dihasilkan oleh *bot* DDA, hal ini berbanding lurus dengan selisih jarak tempuh yang diperoleh pada Gambar 6.3 detik ke-35 dan 42. Detik ke-35 pada Gambar 6.3 menunjukkan *bot* DDA memiliki nilai positif yang berarti *bot* DDA sedang mengungguli *bot hard*, namun pada detik ke-42 terjadi penurunan nilai menjadi negatif, yang mana berarti posisi *bot* DDA dikejar dan diambil oleh *bot hard*, hal ini menunjukkan bahwa *bot* DDA melakukan adaptasi terhadap perilakunya dengan kemampuan dari *bot hard*.

Pada Gambar 6.1 detik ke-42 menuju detik ke-49, garis yang ditunjukkan oleh *bot* DDA mengalami kenaikan sehingga memotong garis *bot hard*. Hal ini terjadi karena pada detik ke-35 dan 42 terjadi perubahan nilai parameter seperti yang dapat dilihat pada Tabel 6.8.

Tabel 6.8 Perubahan Parameter Pengujian 1 *Hard* Detik 35 dan Detik 42

Waktu (detik)	*Caution Angle*	*Steer Sensitivity*	*Max Wander Distance*	*Wander Rate*
35	25,63	0,004	19,061	0,19
42	40	0,008	1	0

Berdasarkan Tabel 6.8, nilai parameter pada perulangan ke-6 termasuk dalam kategori perilaku *Hard*, karena adanya perubahan nilai parameter ini, maka dapat dilihat pada Gambar 6.2 detik ke-42 menuju detik ke-49 terjadi kenaikan kecepatan yang dihasilkan oleh *bot* DDA. Dikarenakan terjadinya kenaikan kecepatan yang dihasilkan tersebut, pada Gambar 6.3 detik ke-42 menuju detik ke-49 dapat dilihat terjadi peningkatan garis grafik, nilai menjadi positif yang berarti *bot* DDA telah mengungguli bot *hard*, hal ini membuktikan bahwa *bot* DDA sudah melakukan adaptasi terhadap perilakunya dan mencoba mengejar *bot* statis. Dari pengujian ini diperoleh rata-rata selisih jarak tempuhnya yaitu 5,41 meter yang berarti jarak kedua mobil berdekatan sejauh 5,41 meter selama balapan berlangsung.

Berdasarkan Gambar 6.4 dan Gambar 6.5 untuk pengujian kedua dan ketiga untuk perilaku *Hard*, kedua pengujian juga menunjukkan grafik yang hampir menyerupai Gambar 6.1. Hasil dari kedua grafik ini sebagai bukti untuk memperkuat bahwa *bot* DDA mampu melakukan adaptasi terhadap perilakunya dengan kemampuan *bot* statis dengan perilaku *Hard*.

b. *Bot* DDA Melawan *Bot Medium*

Berdasarkan hasil dari Gambar 6.6 pada pengujian pertama untuk perilaku *Medium*, terjadi banyak perpotongan garis yang menunjukkan adanya kejar mengejar yang terjadi antara *bot* DDA dan *bot medium*. Pada detik ke-49 sampai detik ke-63 terjadi dua kali perpotongan garis, yaitu pada detik ke-49 menuju detik ke-56, dan pada detik ke-56 menuju detik ke-63. Pada detik ke-49 menuju detik ke-56 terjadi kenaikan jarak yang ditempuh oleh *bot* DDA, dapat dilihat pada Gambar 6.7 detik ke-56 kecepatan yang dihasilkan oleh *bot* DDA di atas angka 120 melebihi kecepatan yang dihasilkan oleh bot *medium*, hal ini menyebabkan selisih total jarak tempuh yang dihasilkan oleh *bot* DDA bernilai positif mengungguli *bot* medium seperti yang tampak pada Gambar 6.8 detik ke-56.

Pada detik ke-56 menuju detik ke-63 seperti yang terlihat pada Gambar 6.6, garis yang ditunjukkan oleh *bot* DDA tampak datar dan hanya mengalami sedikit penurunan, pada detik ini pula terjadi perpotongan garis karena adanya peningkatan jarak tempuh yang ditunjukkan oleh *bot medium*, hal ini terjadi

karena adanya perubahan nilai parameter keluaran seperti yang tertera pada Tabel 6.9.

Tabel 6.9 Perubahan Parameter Pengujian 1 *Medium* Detik ke-49 dan 56

Waktu (detik)	*Caution Angle*	*Steer Sensitivity*	*Max Wander Distance*	*Wander Rate*
49	40	0,008	1	0
56	25,6	0,005	18	0,18

Berdasarkan Tabel 6.9, nilai parameter pada detik ke-49 termasuk perilaku *Hard* kemudian berubah menjadi seperti pada detik ke-56, yang mana nilai-nilai parameter keluaran tersebut masuk ke dalam perilaku *Medium.* Setelah terjadi perubahan nilai parameter ini, kecepatan yang dihasilkan oleh *bot* DDA pun menurun seperti yang dapat dilihat pada Gambar 6.7 pada detik ke-56 menuju detik ke-53. Dengan adanya penurunan kecepatan yang terjadi ini maka dapat dilihat pada Gambar 6.8 nilai selisih pada detik ke-56 hingga detik ke-63 mengalami penurunan menjadi -5 yang berarti posisi diambil oleh bot *medium*. Adanya perubahan jarak tempuh, kecepatan, dan nilai parameter keluaran yang terjadi ini membuktikan bahwa *bot* DDA sudah mampu melakukan adaptasi terhadap perilakunya sesuai dengan kemampuan dari bot *medium*. Dari pengujian ini diperoleh rata-rata selisih jarak tempuhnya yaitu 8,81 meter yang berarti jarak kedua mobil berdekatan sejauh 8,81 meter selama balapan berlangsung.

Berdasarkan Gambar 6.9 dan Gambar 6.10 untuk pengujian kedua dan ketiga untuk bot perilaku *Medium*, kedua grafik pengujian juga menunjukkan banyak terjadinya perpotongan garis yang menunjukkan adanya kejar mengejar antara kedua *bot*. Hasil dari kedua grafik ini sebagai bukti untuk memperkuat bahwa *bot* DDA mampu melakukan adaptasi terhadap perilakunya dengan kemampuan *bot* statis dengan perilaku *Medium*.

c. *Bot* DDA Melawan *Bot Easy*

Berdasarkan hasil dari Gambar 6.11, terjadi banyak perpotongan garis yang menunjukkan adanya kejar mengejar yang terjadi antara *bot* DDA dan *bot easy*. Pada waktu detik ke-63 sampai detik ke-77 terjadi dua kali perpotongan garis, yaitu pada detik ke-63 menuju detik ke-70, dan detik ke-70 menuju detik ke-77. Pada awal detik ke-63 menuju detik ke-70, *bot* DDA memiliki jarak tempuh yang lebih tinggi dibandingkan dengan bot *easy,* namun pada detik ke-70 bot *easy* memotong garis dari *bot* DDA dan mengungguli jarak tempuh yang dihasilkan, hal ini dapat dilihat dari kecepatan yang dihasilkan pada detik ke-63 menuju detik ke-70 pada Gambar 6.12, yang mana kecepatan yang dihasilkan oleh bot *easy* lebih tinggi dibandingkan dengan bot *medium,* hal ini menyebabkan selisih jarak tempuh yang sebelumnya bernilai positif pada detik ke-56, turun menjadi negatif pada detik ke-63 seperti yang terlihat pada Gambar 6.13.

Pada detik ke-70 menuju detik ke-77 pada Gambar 6.11, terlihat garis yang ditunjukkan oleh *bot* DDA mengalami sedikit kenaikan hingga memotong garis *bot*

easy, hal ini terjadi karena pada detik ke-70 terjadi perubahan nilai parameter keluaran seperti yang tertera pada Tabel 6.9.

Tabel 6.10 Perubahan Parameter Pengujian 1 *Easy* Detik ke-63 dan Detik ke-70

Perulangan	*Caution Angle*	*Steer Sensitivity*	*Max Wander Distance*	*Wander Rate*
9	22,5	0,0045	22	0,2
10	27,58	0,005008	20,47	0,19

Berdasarkan Tabel 6.10, nilai parameter pada perulangan ke-63 termasuk perilaku *Easy* kemudian berubah menjadi seperti pada perulangan ke-10, yang mana nilai-nilai parameter keluaran tersebut masuk ke dalam perilaku *Medium.* Setelah terjadi perubahan nilai parameter ini, kecepatan yang dihasilkan oleh *bot* DDA pun meningkat pada detik ke-77 seperti yang dapat dilihat pada Gambar 6.12. Dengan adanya peningkatan kecepatan yang terjadi ini maka dapat dilihat pada Gambar 6.13 nilai selisih yang sebelumnya -7 naik menjadi 9 mengungguli bot *easy.* Adanya perubahan jarak tempuh, kecepatan, dan nilai parameter keluaran yang terjadi ini membuktikan bahwa *bot* DDA sudah mampu melakukan adaptasi terhadap perilakunya sesuai dengan kemampuan dari bot *easy*. Dari pengujian ini diperoleh rata-rata selisih jarak tempuhnya yaitu 7,23 meter yang berarti jarak kedua mobil berdekatan sejauh 7,23 meter selama balapan berlangsung.

Berdasarkan Gambar 6.14 dan Gambar 6.15, kedua grafik pengujian juga menunjukkan banyak terjadinya perpotongan garis yang menunjukkan adanya kejar mengejar antara kedua *bot*. Hasil dari kedua grafik ini sebagai bukti untuk memperkuat bahwa *bot* DDA mampu melakukan adaptasi terhadap perilakunya dengan kemampuan *bot* statis dengan perilaku *Easy*.

d. Penguji Handal Melawan *Bot Hard, Easy,* dan DDA

Berdasarkan hasil dari Gambar 6.18, dapat dikatakan bahwa kemampuan penguji handal mampu menyaingi bahkan menang tipis melawan *bot* dengan perilaku *Hard*, dapat dilihat dari garis grafik yang ditunjukkan mengalami kenaikan dan penurunan yang sangat tipis oleh penguji dan *bot.* Pada Gambar 6.17 dapat dilihat bahwa penguji secara dominan mengungguli *bot Easy* pada hampir setiap perulangan yang terjadi.

Berdasarkan hasil dari Gambar 6.16, garis grafik antara penguji dan *bot* terlihat cukup rapat dan sering terjadi perpotongan, hal ini menunjukkan terjadinya kejar mengejar antara penguji dan *bot.* Dari keseluruhan nilai jarak tempuh penguji dan *bot*, diperoleh selisih rata-ratanya yaitu 2,35 meter, yang mana dapat diasumsikan bahwa mobil *bot* DDA dan penguji berada berdekatan sejauh 2,35 meter sepanjang balapan dan memiliki perbedaan total waktu tempuh yang tidak lebih dari satu detik. Dari hasil ini maka dapat dikatakan kemampuan

penguji mampu menyaingi *bot hard* dan *bot* DDA mampu melakukan adaptasi terhadap perilakunya sesuai dengan kemampuan penguji handal.

e. Penguji Pemula Melawan *Bot Hard, Easy,* dan DDA

Berdasarkan hasil dari Gambar 6.20, dapat dikatakan bahwa kemampuan penguji pemula setara dengan bot dengan perilaku *Easy*, dapat dilihat grafik yang ditunjukkan garis grafik mengalami kenaikan dan penurunan oleh penguji dan bot, hal ini menunjukkan terjadinya kejar mengejar antara penguji dan *bot* dikarenakan memiliki kemampuan yang seimbang. Pada Gambar 6.21, dapat dilihat bahwa garis grafik yang dimiliki oleh *bot* selalu mengungguli penguji kecuali pada perulangan ke-10 hingga 13, yang mana pada perulangan tersebut *bot* lebih dahulu sampai pada belokan yang menyebabkan penurunan garis grafik terjadi. Hal ini menunjukkan bahwa kemampuan penguji berada jauh di bawah kemampuan *bot hard.*

Berdasarkan hasil dari Gambar 6.19, dapat dilihat bahwa terjadi banyak perpotongan antara garis penguji dan *bot,* hal ini menunjukkan terjadinya kejar mengejar oleh penguji dan *bot* yang berarti DDA melakukan adaptasi terhadap perilakunya dengan kemampuan yang dimiliki oleh penguji. Dari keseluruhan nilai jarak tempuh penguji dan *bot*, diperoleh selisih rata-ratanya yaitu 4,49 meter, yang mana dapat diasumsikan bahwa mobil *bot* DDA dan penguji berada berdekatan sejauh 4,49 meter sepanjang balapan dan memiliki perbedaan total waktu tempuh yang tidak lebih dari satu detik. Dari hasil ini maka dapat dikatakan kemampuan penguji mampu setara dengan *bot easy* dan *bot* DDA mampu melakukan adaptasi terhadap perilakunya sesuai dengan kemampuan penguji pemula.

Dari hasil analisis pengujian oleh penguji handal dan pemula, dapat dikatakan bahwa kemampuan yang dimiliki oleh penguji handal jauh lebih baik dibandingkan dengan penguji pemula. Berdasarkan hasil analisis yang dilakukan terhadap kedua penguji, *bot* DDA dapat melakukan adaptasi terhadap perilakunya sesuai dengan kemampuan yang dimiliki oleh penguji. Dengan hal ini maka dapat disimpulkan bahwa implementasi DDA menggunakan *fuzzy* mampu menjawab rumusan-rumusan masalah yang didefinisikan pada penelitian ini.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan-kesimpulan yang diperoleh didasarkan pada rumusan masalah yang sudah dibuat. Kesimpulan diambil berdasarkan hasil dari segala proses mulai dari perancangan hingga pengujian dan analisis. Adapun kesimpulan yang diperoleh adalah berikut:

1. DDA dengan metode *fuzzy* dirancang ke dalam *racing game* menggunakan parameter masukan posisi, jarak, dan durasi *offroad* dan akan merubah nilai dari parameter keluaran yang di antaranya *caution angle, steer sensitivity, max wander distance,* dan *wander rate,* implementasi DDA dilakukan setiap tujuh detik. Pengujian terhadap *bot* statis dilakukan dan dari hasilnya DDA sudah berjalan dengan baik, yang mana untuk pengujian terhadap *bot hard, bot medium,* dan *bot easy* diperoleh rata-rata selisih jarak tempuhnya masing-masing yaitu 5,41 meter, 8,81 meter, dan 7,23 meter.
2. Dari hasil uji yang dilakukan oleh penguji handal dan penguji pemula, terdapat perbedaan dari total waktu yang diperlukan untuk menyelesaikan balapan, yang mana pada penguji handal hanya memerlukan waktu 118 detik untuk menyelesaikan balapan dan rata-rata selisih jarak tempuhnya 2,35 meter, sedangkan penguji pemula memerlukan waktu 126 detik dan rata-rata selisih jarak tempuhnya yaitu 4,49 meter.

7.2 Saran

Meski DDA menggunakan metode *fuzzy* dalam penelitian ini sudah mampu menghasilkan keseimbangan antara pemain dengan *bot*, masih terdapat beberapa kekurangan di dalamnya. Beberapa kekurangan yang bisa diperbaiki adalah :

1. Meningkatkan cara atau proses implementasi DDA, seperti menggantikan proses DDA yang sebelumnya dilakukan tiap tujuh detik menjadi kondisi-kondisi lainnya yang lebih optimal.
2. Memperbanyak parameter masukan dan keluaran yang digunakan dalam *fuzzy*.
3. Menggunakan metode lain sebagai pengganti metode *fuzzy* dalam mengimplementasikan DDA.

DAFTAR PUSTAKA

Arsenault, D., 2009. *Video game* Genre, Evolution and Innovation. *Journal for Computer Game Culture,* Volume 3, pp. 149-176.

Big Fish *Game*s, Inc., n.d. *Game Genres Across the World.* [Online] Tersedia di: <https://www.bigfish*game*s.com/blog/stats/*game*-genres-across-the-world/> [Diakses 12 Desember 2017].

Drope, 2018. *Video Game Genres.* [Online] Tersedia di: <https://tvtropes.org/pmwiki/pmwiki.php/Main/Video*game*Genres> [Diakses 15 Juli 2018].

Fitri, A. & Mahmudy, W. F., 2017. Optimasi Keanggotaan Fuzzy Tsukamoto Menggunakan AlgoritmaGenetika pada Penentuan Prioritas Penerima Zakat. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 1, pp. 125-138.

Gentile, D. A. & Anderson, C. A., 2006. *Video game*s. In: N. J. Salkind, ed. *Encyclopedia of Human Development.* London: Sage Publications, Ltd, pp. 1303-1307.

Intense *Game*, 2015. *[v2.0 BETA] Racing Game Starter Kit - Easily create racing games!.* [Online] Tersedia di: <https://forum.unity.com/threads/v2-0-beta-racing-*game*-starter-kit-easily-create-racing-*game*s.337366/> [Diakses 29 Mei 2018].

Jacobsen, K. S., Olsen, T. & Phan, L. H., 2011. Dynamic Difficulty Adjustment Using *Behavior tree*s. *Master Thesis.*

Kusumadewi, S. & Purnomo, H., 2010. *Aplikasi logika fuzzy untuk pendukung keputusan.* Jakarta: Graha Ilmu.

Porsinelli, P., 2013. *Why is Unity so popular for videogame development?.* [Online] Tersedia di: <https://designa*game*.eu/2013/12/unity-popular-video*game*-development/> [Diakses 15 Juli 2018].

Purba, K. R., Hasanah, R. N. & Muslim, M. A., 2013. Implementasi Logika Fuzzy Untuk Mengatur Perilaku Musuh dalam *Game* Bertipe Action-RPG. *Jurnal EECCIS (Electrics Electronics Communications Controls Informatics Systems),* Volume 7, pp. 15-20.

Rietveld, A., 2014. Circuit-adaptive Rubber Banding. *MSc Thesis.*

Silva, M. P., Silva, V. d. N. & Chaimowicz, L., 2016. Dynamic Difficulty Adjustment Through an Adaptive AI. *Brazilian Symposium on Computer Games and Digital Entertainment,* Volume 14, p. 174.

Skooldays.com, 2013. *Space Race.* [Online] Tersedia di: <http://www.skooldays.com/categories/arcade/ag1133.htm> [Diakses 12 September 2017].

Sutojo, T., Mulyanto, E. & Suhartono, V., 2011. *Kecerdasan Buatan.* Yogyakarta: ANDI.

US *Gamer*, 2015. *The Grandfather of Racing Games.* [Online] Tersedia di: <https://www.us*gamer*.net/articles/the-first-overhead-viewed-racer-was-a-classic> [Diakses 15 Juli 2018].

Wikipedia, 2018. *Microsoft Visual Studio.* [Online] Tersedia di: <https://en.wikipedia.org/wiki/Microsoft_Visual_Studio> [Diakses 17 Juli 2018].